DESIGN OF AN AUTOMATIC MULTI-PROCESSING SYSTEM

RICHARD P. WARRICK

DESIGN OF AN AUTOMATIC MULTI-PROCESSING SYSTEM

*****

Richard P. Warrick

DESIGN OF AN AUTOMATIC MULTI-PROCESSING SYSTEM

by

Richard P. Warrick

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1 9 6 2

DESIGN OF AN AUTOMATIC
MULTI-PROCESSING SYSTEM

BY

Richard P. Warrick

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

from the

United States Naval Postgraduate School

ABSTRACT

As more and more use is made of the computing facilities of the U. S. Naval Postgraduate School, the need for more automation of the facilities is becoming apparent. The addition of a Digital Control Laboratory make a programming system to time share the large scale 1604 computer between this laboratory and operations at the Computer Center desirable. This paper describes a program system to accomplish this. This system will allow closed shop type jobs to be run at the Computer Center and open shop experimentation from the Digital Control Laboratory.

The writer wishes to express his appreciation to Professor Mitchell L. Cotton for his invaluable assistance and encouragement in this project.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

AMPS - Automatic Multi-Processing System

BCD - Binary Coded Decimal

DCL - Digital Control Laboratory

I/O - Input/Output

IR - Interpretive Routine

LM - Local Monitor

MCS - Master Control System

RBL - Relocatable Binary Loader

RRK - Read Remote Keyboard Routine

SM - System Monitor

1. Introduction

This paper will describe an elementary time-sharing system to be used with the CDC 1604 computer at the U. S. Naval Postgraduate School. An interest in a time-sharing system has been prompted by the addition of the Digital Control Laboratory at a remote location from the computer. It is desirable to be able to use the large 1604 computer from this location and to provide for time-sharing when the program from the remote location requires a small percentage of the capacity of the computer.

A perusal of the literature reveals little work in this area being reported on except by Teague /1/ and Corbato /2/ of the Computer Center of MIT. This work allows for any number of remote consoles and makes no restrictions as to program type and size. In order to accomplish this, they have designed special equipment and modifications to the computer (IBM 7090). They make provision for writing programs on an auxiliary store when memory space is needed, calling them back when needed for running. Memory protection among the programs is provided in the hardware.

The system presented here is more specialized in nature and allows for only one remote console. There are several reasons for this approach.

1. The system should be designed for existing equipment with no modifications required.[1]

2. Real time operation from the remote location should be provided for.

3. Relatively simple programming requirements.

By real time operation is meant a reaction time to external stimuli in the microsecond range. This reaction time precludes using any system

---

[1] See Appendix B for a listing of this equipment.

which requires reading a program into memory each time a quantum of computation is to be performed. Since the program in memory cannot be switched, the programs from the remote user are restricted as to type and size.

The other user in this system is a back log of stacked jobs to be run under a monitor system. This user is always considered to have the lower priority.

2. Need for a Monitor System

As more and more use is made of the computing facilities at the U. S. Naval Postgraduate School, the need for more automation of the facilities is becoming apparent. This must be reflected not only in actual hardware but in the manner in which it is used as well.

There are several approaches to this problem. Systems such as Fortran, Jovial, Neliac, involve one method of automating the coding of a program by developing a problem-oriented language as opposed to a machine-oriented language.

Another approach is the use of a supervisory control program or monitor system to process stacks of jobs. Such systems provide various tools to the programmer for aid in input/output and program debugging, but their real merit lies in the fact that such systems are capable of processing more work through the computer than can be processed on an individual basis.

However, no matter what specifications are developed for any system, of necessity certain restrictions will be imposed. In the case of compilers, the restriction is the language itself. In a monitor system, the restrictions may be such things as available tapes or available core storage.

The ultimate goal is to develop a system - in this case a monitor system - which satisfies the requirements of the installation, both from a programming standpoint and from an operations standpoint. Certainly some restrictions and some compromising will be necessary to achieve this goal.

It is felt that the CO-OP Monitor System fills the needs of the installation here with the minimum of restrictions. Since the facilities

are not used primarily for production type runs, some sacrifice can be made in efficiency for flexibility. The aims of the designers of the CO-OP Monitor System are set forth in the introduction to their report of 9 March 1962/3/.

3. Need for a Multi-Processing System

With the installation of the Digital Control Laboratory (DCL) at the Postgraduate School, a need is becoming apparent for a system program which will allow the time sharing of the CDC 1604 computer between closed shop type jobs at the computer site and open shop experimentation at the DCL. For jobs which require only 10 to 20% of the capacity of the 1604 computer, restricting the computer to service the DCL alone is considered wasteful of the computer. To avoid this waste of computing capacity, an Automatic Multi-Processing System (hereafter known as AMPS) has been designed. The remainder of this report will be concerned with the specifications of this system.

AMPS will provide for the following:

1. Automatic control of time sharing between two programs, the Local Monitor (LM) and programs run under it, and a System Monitor (SM) and programs run under it.

2. Operator-Machine communications to both locations.

3. Memory allocation including the sharing of the main memory between the two programs.

4. Linkage to I/O media.

5. Recovery procedures in the event of program failure.

4. System Characteristics

This section will present the overall aims and specifications of AMPS. These specifications will be given in greater detail in Sections 5 and 6.

4.1 Local Monitor

The first program to be considered under AMPS will be the LM. This program will control the operation of the computer from the computer site.

The LM will perform the following functions:

1. Automatic sequencing of jobs as found on the input medium.
2. Loading of program cards and library routines.
3. Linkage to second level control systems.

In most respects, the LM will be similar to the CO-OP Monitor System as described in the CO-OP Monitor System report /3/. (See Appendix A for a summary of the characteristics of this system).

It is desirable to design the LM in such a way as to be compatible with monitor operation when the computer is not shared with the DCL. If this can be achieved, the programmer will not know whether his program is run with time sharing or not. It is apparent, however, that some reduction in memory size will be required since the computer must hold a larger system program when in the time sharing mode. The on-line peripheral equipment will have to be restricted also.

4.2 System Monitor

A second program to be considered under AMPS will be the SM. This program will control the operation at the remote location.

The SM will perform the following functions:

1. Call programs from the standard library tape or any other tape,

including the remote 160 computer, under control of the remote console.

2. Execute these programs on command.

3. Execute these programs in an interpretive mode either full speed or in a "stepping mode".

4. Provide the simulation of the computer console on the remote display. This routine will automatically be called whenever a program is completed or after each step when in the interpretive mode.

5. Provide for the inspection and changing of any selected cell in the computer within that portion of memory assigned to the SM.

The call routine contained in the SM will merely be a linkage to the load routine to be contained in the LM. This routine will be told to load programs called into that portion of core reserved for SM use.

It is anticipated that two basic types of programs will be run under the SM. These will be checked-out programs used to aid in other development work and programs coded by novice programmers and unchecked-out.

The checked-out programs will be considered first. It is anticipated that many of these will be used to investigate real time processes. Most of them will not require more than 3000 to 4000 words of core storage nor more than 10% of the computing capacity of the computer.

Since these programs will depend on real time external inputs, the timing for inputs cannot be set in advance but must be placed under interrupt control. Because these interrupts are unpredictable, care must be taken in the design of LM programs to insure against program errors caused by them. This restriction will be covered in detail later.

In order to allow efficient use of the computer and at the same time allow for "debugging at the console" and student instruction, it appears desirable to provide for an interpretive program to execute programs

from the remote console. By using an interpretive program, the danger of an unknown program destroying LM programs is avoided.

This program should contain provisions for running the program "full speed" or in a "stepping mode". By full speed is meant the rapid execution of the instructions but remaining in the interpretive mode. By stepping mode is meant an action similar to that produced by the "step" switch on the main console. In the full speed mode, provision should also be made to set "break points".

To further simulate the main console, each time the program being interpreted is halted, the contents of the psuedo registers should be displayed on the remote console.

5. Problems of Multi-Processing

There are many problems that may occur in a multi-processing system. The three discussed here (memory assignment, I/O control, and switching) are considered the most important. Others will be discussed within the details of the monitors.

5.1 Memory Assignment

One of the major problem areas in designing a time sharing system is that of safe guarding memory. The system will not be useful if programs run under the LM destroy programs in the SM and vice versa.

It would be desirable to provide the system with the capability of automatically adjusting the size of the memory assigned to the various programs to that needed at the time. With the present state of the art, there appears to be no simple solution to this problem of dynamic storage allocation. The programs expected to be run under the SM are of a restricted nature, however, so that an allocation of a fixed portion of memory to the SM seems reasonable. These SM programs are considered to be limited to 10 to 20% of the capacity of the computer. It is also felt that a first system can more readily be made operational by assigning a fixed area of storage to the system programs.

Even with this scheme of separating the memory, it is almost certain that programs will overlap occasionally and cause trouble. The system design will attempt to minimize the chances of this occurring.

5.2 I/O Control

Most of the control functions connected with program switching will be performed under interrupt control. Since these interrupts are unpredictable, all programs must be coded in such a way that if they are

interrupted, the program will not be destroyed. One major area of concern here is in input/output since some routines are susceptible to damage by interrupts.

One particular example of this is the input routine contained in SCRAP. Since only the first two words read in to memory are needed to commence processing, a section of the read routine inspects the control cell (cell 1, 3, or 5 for input) in the computer until it indicates that these two words have been received. If this routine were interrupted during this inspection, the buffering would continue and when a return to this routine is made, the proper check would have been missed causing the routine to hang-up in a loop.

Any problem here can be avoided if all programs use the standard routines provided in AMPS. These routines will be coded such that their being interrupted can be tolerated.

It is also desirable to isolate the I/O channel usage between programs. If an I/O channel is shared between two programs, problems will arise in timing, particularly in the matter of error checking. One program may initiate an input/output operation and when it is completed, the other program may do the same but before the first program has a chance to check for errors. By the time the first program does check, any pertinent information has been lost.

5.3 Switching Between Programs

The philosophy of switching between the LM and the SM will be that the LM has the use of the computer except at those times when a need is indicated by the SM. This need will, in general, be indicated by an interrupt. When this occurs, the SM will be given control and control

will remain with the SM until it completes its task. When control is given to the SM, all conditions of the computer must be saved for future use by the LM. This includes removing any interrupt selections for interrupt conditions that the SM or programs run under it are able to satisfy. This is true even though all further interrupts will be locked out during the period that the SM has control since their occurrence is sensed as soon as the interrupt routine exits through cell 7. When the SM has completed its task, all registers and conditions will be restored and control will be returned to the LM program at the point where it was interrupted.

With this method of control, any discussion of efficiency is not relevant. It will appear to the user at the remote console as if he is the only user. The full uninterrupted use of the computer will be available to him whenever it is called for. The efficiency of operation of the LM and its processing of stacked jobs will be simply the percentage of time not taken by the SM.

6. Local Monitor Restrictions

The LM will perform the functions of the CO-OP Monitor System with few restrictions. The portion of memory assigned to the SM will be immediately after the resident portion of AMPS (see Figure 1). Programs run under the LM will have available all memory from the end of the SM portion to the high end of memory. The following sections discuss several restrictions on the programs run under LM control. Specific modifications to the CO-OP Monitor System necessary are described in Section 8.1.

6.1 Size of Programs

The LM programs will of necessity be limited in size from that permitted with normal monitor control. This is necessary for two reasons.

First, a larger resident portion of the monitor must be in cores at all times. The larger size of this resident portion is required since the monitor is now controlling programs from two locations and must contain routines to accept control statements from these sources.

Second, a portion of the memory must be set aside for SM use. Since no dynamic storage allocation is used, this block reserved for SM use will necessarily be of such size as to hold all programs that are expected to be needed. It will be noted that the remote console has the only control over this block size. Since the SM portion of memory is located at the beginning of available memory, a change in this size will necessitate moving the low end of memory available to the LM up. The details of this will be given in Sections 7.3 and 8.1.6. If the remote location specifies a block of such size that LM programs will not fit, the jobs requiring these programs will have to be scratched for a later run. It is expected that considerable useful work can be accomplished even with this restricted memory size.

By assigning the LM portion of memory to the highest portion of memory, the load routines and systems programs may be used without change. This is true of systems programs that assign and use high memory down for "common" storage and ones that have "open end tables" since these should all contain an end of memory cut-off. The working storage assigned to assemblers and compilers is automatically adjusted to that available at the time without the use of any parameters.

6.2 Program Type Restrictions

Program failure is a real danger to any multi-processing system. If programs running under the LM or SM fail when the computer is running under AMPS control, all processing is lost until the operator takes the proper recovery measures. Even after recovery has been accomplished, the jobs being processed by either or both the LM and SM may be damaged in such a way that they must be reloaded or just as fatal, essential information may be lost during the recovery period.

It is expected that systems programs will be designed and checked-out in such a way that they will run to completion and not use any portion of memory not assigned. This should be true even if the information on which they must operate is faulty. This will, in general, also be true of compiler generated programs. If a compiler generates a program at all, it is expected that the only error effecting AMPS operation which may occur is excessive running time. This error is guarded against somewhat by the time limit indicated on the job description card.

There appears to be no way to inspect machine coded programs to ascertain their running characteristics without actually running them. It is expected that block storage allocation specified by the programmer will be indicated in the object program. There is no guarantee, however,

that this is all the storage used by the program. There is also the chance of programmer error in assigning symbolic addresses and errors in index register use. In order to keep the number of program stoppages to a minimum, it appears reasonable to bar unchecked-out assembly derived programs from being run under AMPS. The decision as to what constitutes a checked-out program will obviously be up to the programmer and the staff member accepting programs to be run under AMPS. Assemblies themselves can be made, however, since memory use is controlled by the assemblers and they will be restricted to the LM portion of memory.

The AMPS system should be able to accept programs generated by other systems without restriction except for size as mentioned above. This may vary, of course, depending on what programming systems are included in AMPS.

6.3 I/O Assignments

It is planned to make extensive use of the satellite system for LM input/output. There are several reasons for this choice.

First is the limited equipment available. Only two 1607 tape subsystems are anticipated and one of these will be required by the SM. Only four tape units are thus available for the LM. This is insufficient for most LM operations. To increase the number of tapes available, the 160 computer in conjunction with the 163 tape units will be utilized. It is anticipated that the 160 will have a 163 tape subsystem with two tape units.

It is also felt that the use of the 160 computer can speed up input/output without use of complicated buffer schemes which may be damaged by interrupts. The 160 can read a record from the tape and store it intern-

ally until called for by the 1604. The transfer to the 1604 proceeds at high speed since it does not depend on tape motion. The same is true of output from the 1604. If the 160 is used to write the output tape and to drive the on-line printer, the output from the 1604 will not depend on tape motion. Of course, more efficient operation will result if a 160A computer is used as the satellite computer in place of the 160.

By using the 160 as the standard input medium, the facility to modify the input stream without interfering with the operation of the 1604 is gained. This feature is useful if a high priority job is to be run during the sequencing of a long input tape.

It is not anticipated that the card reader/punch will be used extensively on-line but it will be available if required.

The following channel utilization is planned for the LM.

1. All control information to the LM will be via the console typewriter on channel 1.

2. All error information and on-line comments will be via the on-line printer. The 160 computer will be used to drive this printer. The information will be transferred to the 160 computer via satellite on channel 4.

3. Input to the LM will be from the 160 computer via satellite on channel 3. The 160 will pick up the information from the system input tape on the 163 tape subsystem.

4. The listable output will be to the 160 computer via satellite on channel 4. The 160 will write the information on the 163 tape subsystem.

5. The LM library and binary output tapes will be assigned to 1607 tape units on channels 3 and 4.

6. Two 1607 tape units on channel 3 and 4 will be available to the LM as scratch units.

The standard entries in the RHT[2] of the LM will have the following meanings. This table is also used by the SM.

00 - System library (Channel 3-4 1607 tape #1)

50 - Monitor input (Local 160 in satellite)

51 - Monitor output (Local 160 in satellite)

52 - Monitor punch (Channel 3-4 1607 tape #2)

53 - Input comment (Console typewriter)

54 - Output comment (Printer)

55 - Accounting medium

56 - Standard scratch #1 (Channel 3-4 1607 tape #3)

57 - Standard scratch #2 (Channel 3-4 1607 tape #4)

58 - SM input (Remote 160 in satellite)

59 - SM output (Remote 160 in satellite)

60 - SM scratch (Channel 5-6 1607 tape #2)

61 - SM library (Channel 5-6 1607 tape #1

62 - Remote display output

63 - Memory is the I/O medium

6.4 I/O Restrictions

It is expected that the I/O routines will remain as described in the CO-OP Monitor report /3/ except that the routines themselves will be modified if necessary to preclude any interaction with the interrupts. It will not matter whether the systems programs such as FORTRAN, SCRAP, etc.,

---

[2] See Section A3.2 for description of this table.

use these routines as long as their input/output routines are capable of being interrupted at any time. It is felt, however, that it would be best if they did since the flexibility of equipment assignments is gained very simply in this manner.

Since the standard input/output medium for the LM will be the 160 computer in satellite, all programs should use the standard routines for communication with the standard input and output tapes. This will be necessary because the 160 computer must have a program compatable with the companion routine in the 1604 computer. The use of the standard routines will avoid any concern by the programmers over proper programming for the satellite system. These routines may be used just the same as the ones communicating with magnetic tape.

The assemblers and compilers to be used with the LM will have to contain provision to set up the proper linkages to the routines.

7. System Monitor Details

The SM will perform the functions outlined in Section 4.2. In order to perform these functions, several routines will be included in the resident portion of AMPS or be available on the standard library tape. The resident routines peculiar to the SM will be Read Remote Keyboard, Load Program, and Display and Enter routines. A description of these routines is given in the following sections. Any number of routines for SM use may be added to the library tape. The only one described here will be the Interpretive routine. Of course, all library programs included on the library tape for LM use may also be called by the SM.

7.1 Resident Routines

The following routines will be included in the resident portion of AMPS.

7.1.1 Read Remote Keyboard Routine

A standard routine is necessary to Read the Remote Keyboard (RRK). This routine will be used to assemble all information received from the remote keyboard and take action as indicated. Since an expanded keyboard is available, two special keys will be defined which will not interfere with normal interpretation of standard keyboard symbols. These are an ENTER key (represented in this report by a period) and an ERASE key. The comma will be used to separate parameters.

Each key on the remote keyboard, when depressed, will produce an interrupt in the computer. An input message will be read a character at a time, the presence of a new character being indicated by an interrupt from the keyboard. These characters will be assembled (in BCD code) into a word until a comma or ENTER is encountered unless the first char-

acter is a numeric. If a comma is encountered, the next word will be used to accumulate the characters. The commas will be deleted.

If the first character of a parameter is a numeric, these will be stored until a comma (or ENTER) is received. The last character will then be inspected to see if it is the octal indicator (small 8 as defined in Neliac). If it is, the number will be simply converted to binary and stored in the next parameter word. If the last character is not the octal indicator, the number will be converted from decimal to binary fixed point and stored in the next parameter word. It should be noted that up to 16 octal digits may be given in one parameter. If less, they will be right justified in the parameter word. Any decimal number given must be less than $2^{47}-1$. Any parameter starting with a numeric and containing other than zero through nine will produce an error indication on the remote display.

It should be noted that there is a limit of 8 BCD characters to any one parameter. It is felt that this is sufficient for routine names and parameters.

The ERASE key will be used to reset the RRK routine so that it will be ready to begin receiving a new message. No action will be taken on the information already received.

When an ENTER is received, the first word will be compared to a table of available routines to determine if this routine is available. If so, a jump to the address of this routine will be made. If not, an indication will be made on the remote display. It should be noted that the same names may be used by the SM routines and the LM routines as the table of names are completely separate. It would appear desirable to have a common table of routines available but if this is done, there is

too much danger of the SM executing a routine already being executed by the LM. One LM routine will be used by the SM but it will contain a safeguard so that this cannot happen.

The characters will also be transferred to the remote display as received. The operator may thus inspect his message as it is being entered. If an error is found, the backspace key may be used. This will reset the RRK routine so that it will load the next character in place of the previous one. Any number of backspaces may be given in succession but all characters after the one replaced will be deleted.

Blank characters will be ignored except that they will be displayed as blank characters. Each assembled word will not have any imbedded blanks.

In summary, the form of the message to RRK will be:

ROUTINE,PARA1,PARA2,.......PARAn(ENTER)

When the routine named is executed, a return jump will be made to the address given in the available routines table. The address of the first parameter word will be in the A register on entrance. All further interrupts from the keyboard will be removed until the routine being processed is finished. This is necessary to avoid the problem of multiple interrupts.

If the ENTER key is depressed without entering a new message, the previous contents of the input buffer will be used. In this manner a routine may be executed repeatedly without the need for retyping the message.

The input messages will use successive lines of the display up to a maximum of 62. After 62 have been received, the next message will be displayed on the first line replacing the one there previously. A CLEAR

command will clear the display of all messages from SM resident routines. A DISPLAY command will also clear those lines needed to display the information called for.

Details and flow charts of this routine are given in Section 8.2.1.

7.1.2 Load Programs Routine

For this routine, the services of the Relocatable Binary Loader (RBL) contained in the LM will be called on. The proper linkage to the RBL will be set up and then RBL executed. There is a danger here that RBL may be executed by the SM at a time when it is in the process of being executed by the LM. This case will be guarded against by the proper use of a warning flag. This flag will be set whenever the LM enters RBL and reset when an exit from RBL is performed. The SM routine will interrogate this flag before it executes RBL. If it is set, the SM will change the exit address of the RBL to a SM routine so that when the RBL has completed its present task, control will be returned to the SM routine rather than the calling LM routine. The SM will then proceed with the loading operation after which control will be returned to the proper point in the LM routine.

The program called will be loaded into that portion of memory reserved for the SM. If the SM portion is full, an indication of this fact will be given on the remote display. The operator can then either reset the size of the SM portion of memory, execute a FREE operation or scratch the job on hand. In either of the first two cases, the call will have to be given again.

The following keyboard commands will be available to control the load routine.

1. LOAD,NAME,X. This command will cause program NAME to be loaded from input medium X, where X is the RHT table entry location for the input device. It should be noted that only entries in the range 50-63 will be considered by this routine. A table of meanings of these entries can be found in Section 6.3 of this report. NAME may be any name up to 8 characters in length. The input medium will be scanned until a name is found matching that given. If none is found, this fact will be indicated on the remote display.

2. HOLD. This command protects programs presently loaded from being destroyed by a new LOAD command. Normally, a new program will be loaded over any old programs in memory unless the HOLD command is given.

3. FREE. This command removes the effect of any HOLD commands that may have been given.

4. SIZE,X. This command sets the size of the SM portion of memory to X words.

The HOLD, FREE, and SIZE commands will all operate to control the LOAD routine somewhat indirectly. The LOAD routine uses the RBL routine of the LM resident to load programs. One parameter sent to this routine is the start and end of available memory (MEM). RBL returns the new start and finish of available memory (NMEM). With this in mind, these commands will be explained in somewhat more detail.

HOLD will substitute the last NMEM perameter for MEM. Hence, the next time a LOAD command is given, the previously loaded programs will be protected.

FREE will substitute a constant, IMEM, for MEM. IMEM will contain the end of the resident portion of AMPS as the starting address. The end address will be some increment plus the starting address. It is expected

that the initial increment will be on the order of $10,000_8$.

SIZE will change the end address of IMEM so that it is the end of the resident portion of AMPS plus the increment given in the SIZE command. The effective starting address will not be changed by this command. A FREE command must be given for this new size to become effective. The effect will also be delayed until the LM starts a new job.

Another note of caution. The LM use of the RBL routine leaves the standard library tape positioned for most efficient use. A call from the standard library tape by this routine will disturb this positioning. This routine will rewind the tape before and after a call is made from it. This will mean that the search time of the library tape by the LM will be increased in certain circumstances but operation would proceed without error.

The entry point names and their corresponding addresses will be entered in a table of available routines. This table will be available to the RRK routine so that it may execute the loaded routines. Permanent entries will be in this table for those routines forming part of the SM and routines in the LM used by the SM.

Details and flow charts of this routine are given in Section 8.2.2.

7.1.3 Display and Enter Routine

A routine will be provided in the SM to display and enter information in any cell in that portion of memory set aside for SM program use. This routine may most easily be described by defining the commands accepted by it. These are:

1. DISPLAY,XXXXX,YYYYY,R. This command will cause the contents of cells XXXXX to YYYYY to be displayed on the remote console. XXXXX and

YYYYY may not differ by more than 124. The R parameter is present if XXXXX and YYYYY are to be considered relative to the start of that portion of memory assigned to the SM. Non-significant zeros need not be entered. This routine will be effective over all memory when the R parameter is missing. If the YYYYY parameter is missing, only the XXXXX cell will be displayed. As described in Section 7.1.1, XXXXX and YYYYY may be either octal or decimal.

If the ENTER key is struck again without a new message being entered, a block of memory the same size as specified originally will be displayed, this time starting at cell YYYYY. If YYYYY was not given originally, the cell after XXXXX will be displayed. The address of each cell displayed will be displayed also.

Details and flow charts of this routine are given in Section 8.2.3.

2. ENTER,XXXXX,O,O,......O. This command will cause the 16 digit numbers (O) to be entered into the cells starting at XXXXX where XXXXX is defined as in the DISPLAY command. If this command is given after a DISPLAY command, the new contents of XXXXX will be displayed along with any other cells requested. This command will only be effective over that portion of memory assigned to the SM. Up to ten O parameters may be given at one time.

Details and flow charts of this routine are given in Section 8.2.4.

7.1.4 Summary of Resident Commands

1. CLEAR. Removes all messages from the remote display.

2. LOAD,X,NAME. Loads program "NAME" from medium X.

3. HOLD. Prevents new LOAD command from erasing presently loaded programs. New program will start from end of present programs.

4. FREE. Resets LOAD routine so that the next program loaded will start at end of resident portion of AMPS.

5. SIZE,X. Sets the size of the SM portion of memory to X words.

6. DISPLAY,XXXXX,YYYYY,R. Causes contents of cells XXXXX to YYYYY to be displayed.

7. ENTER,XXXXX,0,0,......0. Causes number (0) to be entered in cells starting at XXXXX.

7.1.5 Satellite Read/Write Routine

A satellite read/write routine for the remote 160 computer will be included in AMPS resident. It is expected that most SM input/output will be via this routine. See Section 8.1.7 for details of this routine. The name and address of this routine must be in AEDNT[3] since it is used by the RBL.

7.2 Interpretive Routine

The Interpretive Routine (IR) will be used for assembly derived programs. These programs will be assembled under control of the LM or the full monitor. The object program from the assembly will be the input to the interpretive routine. This object program can be loaded from any of the available equipment by using an internal routine in the IR.

After the program has begun, it may be "stepped" through by specifying the stepping mode to the IR. Each time the ENTER key on the remote console is struck, the program will be advanced one instruction. During each halt, the contents of the psuedo registers will be displayed, and control will return to the LM.

---

[3]See Section A3.2.

The program may also be run without interruption in the interpretive mode by specifying the full speed mode. A break point may be set which will cause a halt before that instruction specified by the break point is executed. The stepping mode may be specified at this point or a new break point set and/or the ENTER key struck again. The contents of the psuedo registers will be displayed after each halt. EXF commands will be ignored.

The following keyboard commands will be available to control the IR.

1. IRLOAD,X. This command will pick up a program from input medium X, where X is the position of the input equipment in the RHT and store it in memory ready for execution by the IR.

2. STEP. Sets up the IR so that each time the ENTER key is struck, the program being interpreted will be advanced one instruction.

3. RUN. Sets up the IR so that striking the ENTER key will cause the execution of the program to the end or to a break point if one is set.

4. BREAK,XXXXX,XXXXX,......XXXXX. Sets up the IR so that a program will halt at XXXXX if a RUN command is given. XXXXX is relative to the start of the program. As described in Section 7.1.1, XXXXX may be either octal or decimal. Up to 5 break points may be set at one time. These break points will remain set until a new BREAK command is given or an IRLOAD or STEP command is given.

7.3 Program Size Restriction

As mentioned previously, it is expected that most SM programs will not exceed 3000 to 4000 words of memory. This appears to be sufficient for many of the programs anticipated at this time. To increase the size of the SM portion to a figure much greater than this will preclude run-

ning many LM programs and therefore no advantage will be gained from the use of the AMPS.

This portion of memory, however, will be variable from the remote keyboard. This will be done by use of the SIZE command. When this command is given, the memory recorder contained in the LM will be changed so that the available memory is further restricted. When the next program is called in by the LM, this new memory size will be used to check if it will fit. It should be noted that although LM system programs will be modified to fit in a restricted memory, this command has no effect on the size of memory required to hold them.

One real problem with this command is what to do about a program being executed at the time the size of the SM portion of memory is changed. It appears that the easiest way at present to handle this is to delay the effect of the SIZE command until the LM is ready to start on a new job. It is hoped that the DCL operator will exercise some discretion in arbitrarily increasing the size of the SM portion of memory. If this option is abused, the usefulness of the AMPS will be greatly decreased.

The two other commands available which indirectly control the size of the SM portion of memory are the HOLD and FREE commands. HOLD is used to prevent the LOAD command from loading the next program called from destroying previously loaded programs. The FREE command need only be given after a HOLD and it resets the loader so that the next program called will be loaded from the beginning of the SM portion of memory.

7.4 Interrupt Control

One of the design criterion of AMPS is to provide features so that real time processes could be investigated. Because the inputs to programs

used to investigate these processes are real time in nature, their presence cannot be anticipated by the program, and this presence must be communicated in some manner. It is expected that this will be done by use of the interrupt feature of the 1604 computer as amplified by the interrupt package supplied with the CO-OP Monitor System.

7.5 Equipment Assignments

It is planned to communicate with the remote console via channel 7 of the 1604 computer. This choice of channel may appear arbitrary but it is predicated on the following reasoning.

It appears very desirable not to have the SM and LM share the same channel. This will avoid many difficulties connected with two programs time sharing a channel. Since three of the I/O channels are buffered, separation of channel assignments mean that their use by the different programs is completely independent. Two channels will be required for the LM. Channels 1 and 2 are normally assigned to the console equipment and 3 and 4 will be required for the 160 satellite system and the 1607 tape subsystem assigned to the LM. This only leaves 5 and 6 for SM use. These channels will be required for the 1607 tape subsystem assigned to the SM. This 1607 is necessary to the SM for satellite communication with the remote 160 computer. It is also very desirable for communication with the remote location to proceed simultaneously with communication with the remote console. Hence the decision to assign the remote console to channel 7.

Although channel 7 is unbuffered, the efficiency of the 1604 computer should not be effected materially. Transfer from the 1604 to the remote display is memory to memory and should proceed rather rapidly. This information is also for human consumption, hence the data rate cannot be too

high. Transfer from the remote display to the 1604 will originate from the keyboard and will be under interrupt control. The input to the computer should be ready to be sensed by the time the input instruction is given. Since this input is character by character with an interrupt for each character, no waiting for information will be required. Hence, input should not effect efficiency noticeably. Since this information originates at the keyboard, the data rate here is also low.

To summarize, the following channel utilization is planned for the SM.

1. Satellite communication to the remote 160 computer via channels 5 and 6.

2. Operator-Monitor communication to the display unit via channel 7.

3. Other casual inputs will be via channels 5 and 6.

4. SM library and output tape will be via channel 5 and 6 tape units.

8. Programming Details

This section contains details of the programming necessary for the basic portions of AMPS. It is felt that these are sufficiently detailed to fully understand the routines but leave enough latitude for individual programmers to exploit their skill.

8.1 CO-OP Monitor System Modifications

The following sections contain brief descriptions of the modifications necessary to the CO-OP Monitor System to make it compatible with the overall system. The description of these modifications is necessarily brief and sometimes sketchy since few details of these routines are available. It is felt that the intent of these changes are clear enough so that when the details are available, it will not be difficult to find the areas requiring changes.

8.1.1 Equipment Assignment

It is expected that a parameter in the Memory Recorder section of this routine can be changed to move the initial low point of available memory as required. This parameter will be under the indirect control of the SIZE routine in the SM. The actual changes to this parameter will be supplied by the Bootstrap Loader. This is necessary in order that the LM does not load programs into the SM portion of memory. It is assumed that this parameter will be in a fixed location so that it will always be known to the Bootstrap Loader.

8.1.2 Clock Control

Since it is not desired for the time spent in the SM to be charged to a program running under the LM, the real-time clock must be saved on

entry to the SM and the interrupt on clock overflow removed. When the SM exits back to the LM program, the clock must be reset and the interrupt restored. No specific modifications to this routine should be necessary.

8.1.3 I/O Package

No changes should be necessary to this routine. A routine to read and write using the satellite system must be added to the system library, however, a description of this routine is given in Section 8.1.7.

8.1.4 Interrupt Package

A permanent section must be added to this routine to recognize the interrupts from the remote keyboard. The LM will not be able to clear this interrupt. When this interrupt is detected, the Read Remote Keyboard routine will be entered and further interrupts will be locked out until a return is made from the SM program. The routine should be left so that other sections operate as described. It will also be necessary for this routine to be contained in the permanent part of MCS and not be reloaded after each job. The SM will use this routine and if it were to be reloaded, information would be lost.

8.1.5 Relocatable Binary Loader

This routine is used by both the LM and the SM. To prevent interaction in the event that the LM is executing the routine at a time that it is called for by the SM, a flag (called RBLFLAG in the SM LOAD routine) must be added which is set whenever the LM enters the routine and cleared when it exits. It is assumed that this routine is in a fixed position in memory so that its location will be known to the SM. It is also assumed that the memory usage by this routine is completely controlled by the

parameter given it on entrance. It is on the basis of these assumptions that it is feasible for both the LM and SM to use the routine.

8.1.6 Bootstrap Loader

This routine must be revised so that the loading of MCS and/or the recovery routine will not destroy that portion of memory assigned to the SM. As noted previously, the SM portion of memory is immediately after the resident portion of AMPS (MCS) but before the transient portion of MCS. Thus the permanent portion of MCS must be loaded lower in memory than the SM portion of memory but the transient portion after the SM portion. See Figure 1 for a memory map showing this layout. This routine must also inspect the current SM memory available parameter supplied by the SIZE routine in the SM. This parameter is called BMEM. If this parameter has been changed since the last bootstrap load, the new available memory size will be given to the Memory Recorder section of the Equipment Assignment routine so that this new size may be used to control the loading of programs for the next job. It will also cause a message to be displayed that the new size is effective and change the limits in MEM. The new LM limits are BMEM + 1 to the old end and the new SM limits are given in BMEM.

When the system is initially loaded by use of the "bootstrap" sequence, this routine must also load the resident portion of the SM into memory.

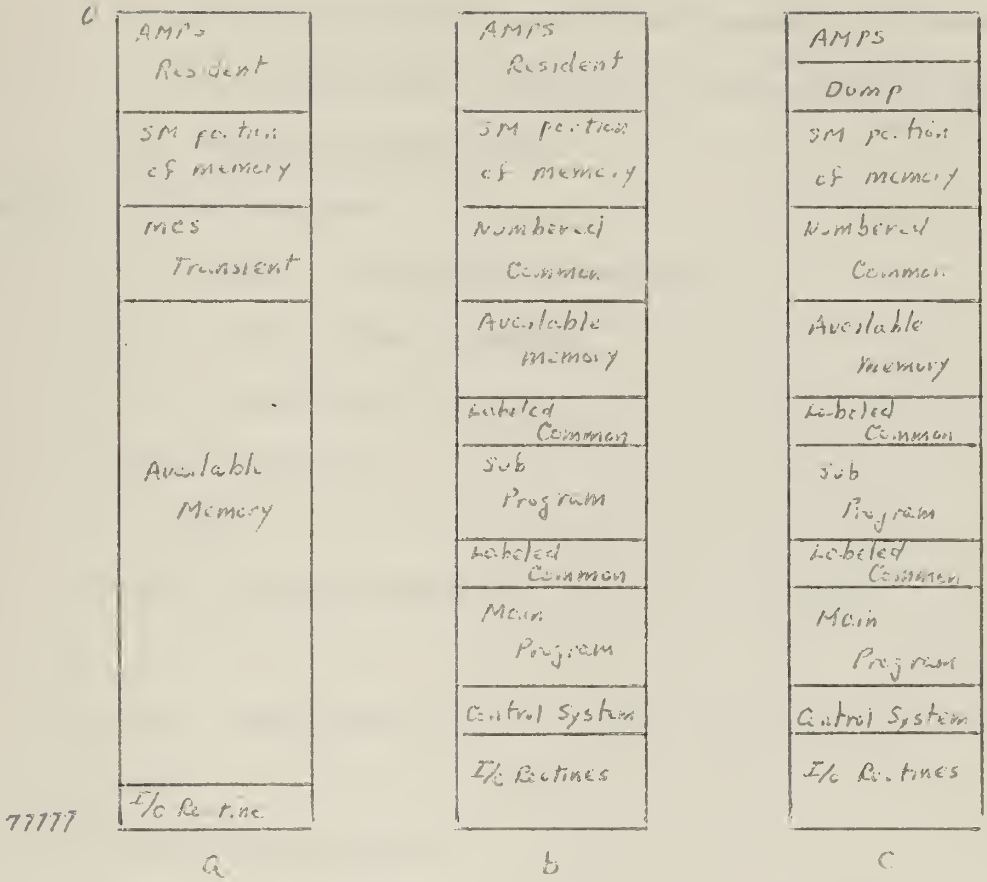

Figure 1. Memory usage under AMPS. a - Initial, b - During program execution, c - During "abnormal" termination when full dump is called for.

8.1.7 Satellite Read/Write Routine

This section follows the form of the CO-OP Monitor System report /3/.

TITLE: Satellite System Read/Write

PURPOSE: To transfer one record to or from the 160 computer in satellite. To perform all auxiliary operations necessary for general satellite input/output such as check for errors, rewind, skip forward, and write end-of-file mark.

USAGE: Calling Sequence:

```
        ENA       Loc Interrupt Subroutine
L       RTJ   0   Name of routine
        R.M.  0   E.C.
L+1     F.C.  0   A
        I         B
L+2     Alternate Return
                  E
L+3     Normal Return
```

where

R.M. is not required

E.C. = Equipment Code (4 octal digits, $D_1 0 D_3 0$)

D_1 = Channel D_3 = 0 for tape

= 1 for printer

I = Interrupt Selection = 0 without interrupt

= 1 with interrupt

F.C. = Function Code explained in detail below

A,B,E, detailed under each function code.

Functions - In all cases if a parameter error is encountered, E is set equal to 0 and control is transferred to the error return. Parameter

errors are such things as channel no. not equal to 3-4, function code missing, etc.

1. Read/Write Only. - F.C. = 1

   Starts transmission of data to or from buffer specified by A = first address, B = last address + 1, then exits. If I = 1, selects the appropriate interrupt before exiting. When the interrupt occurs, control is transferred to the routine whose location has been specified in the A register.

2. Check Only. - F.C. = 2

   Checks previous transmission of data on specified channel. If an error has occurred, E is set equal to 1 and control is transferred to the alternate return. In all other cases, control is transferred to the normal return. For end-of-file mark encountered E = 4. If a combination of conditions has arisen E is set equal to the sum of the individual E's.

   In all cases the routine sets the channel inactive and leaves the number of words transmitted in the accumulator.

3. Read/Write With Checking. - F.C. = 3

   Retains control until data transmission and checking as described above have been accomplished.

4. Sense Equipment Ready. - F.C. = 4

   Determines whether specified equipment is in ready status. If not ready status E = 1 and control is transferred to alternate return. If ready status, control is transferred to normal return.

5. Skip Forward B Records. - F.C. = 5

   When an input channel is specified, skips forward the number of records specified by B. If I = 1 selects the appropriate interrupt

before exiting. When the interrupt occurs, control is transferred to the routine whose location has been specified in the A register. No operation is performed when an output channel is specified.

6. Skip Forward Past End-of-File or Write End-of-File Mark. - F.C. = 6.

   When an input channel is specified, skips forward until an end-of-file mark is encountered. When an output channel is specified, writes an end-of-file mark.

7. No Operation. - F.C. = 7

   No operation is performed.

8. Rewind With Interlock. - F.C. = 8

   Rewinds the specified tape with interlock.

9. No Operation. - F.C. = 9

   No operation is performed.

A flow chart for this routine is shown in Figure 2 on the following pages. These should be self explanatory. FLAG I and FLAG II refer to satellite communication flags I and II. Other symbols used are defined below:

CBF	- Channel Busy Flag. Set indicates channel is busy.
BUF	- This is the name of a dummy cell used for a one word transfer to the 160 in order to communicate the action required.
ACT	- Table of destination codes used by the 160 to determine the destination of transferred information. ACT(0) contains code for tape ACT(1) contains code for printer
ACTA	- Table of action codes used by the 160 to determine action to perform. ACTA(0) contains code to rewind write tape ACTA(1) contains code to rewind printer ACTA(2) contains code to rewind read tape
ACTB	- Table of action codes used by the 160 to determine action to perform.

ACTB(0) write E.O.F. on write tape
ACTB(1) write E.O.F. on printer

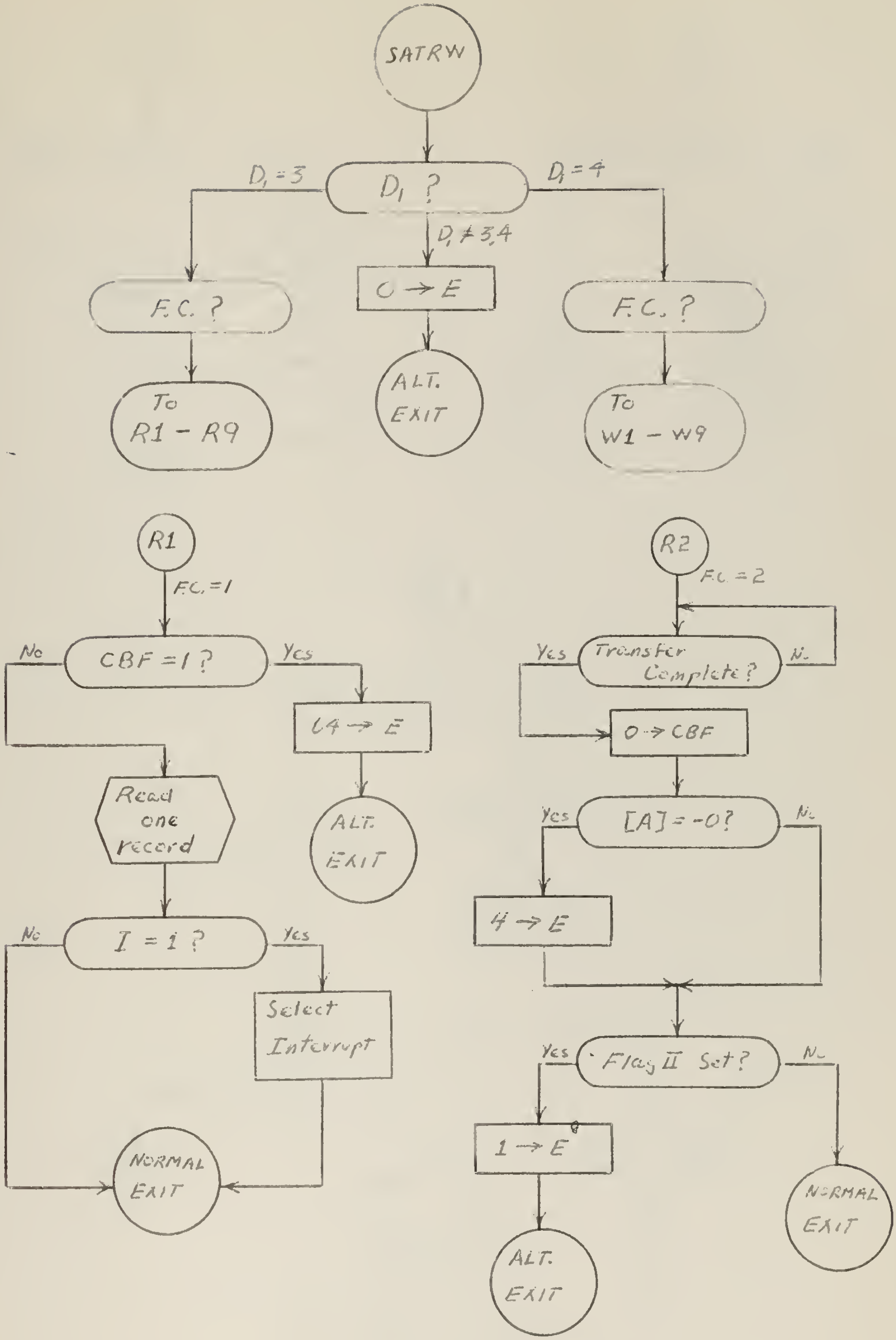

Figure 2. Flow chart of Satellite Read/Write routine.

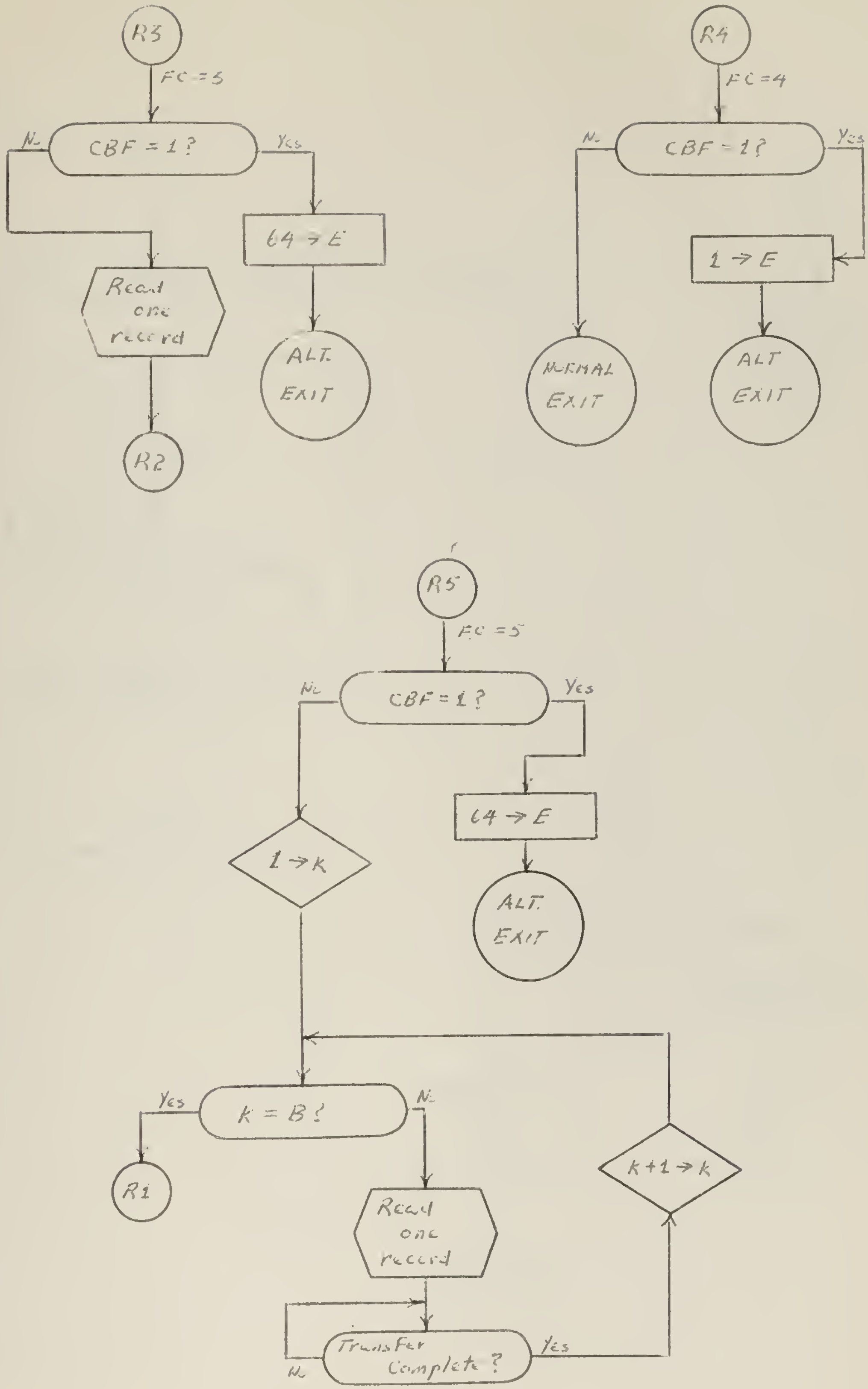

Figure 2. (Continued)

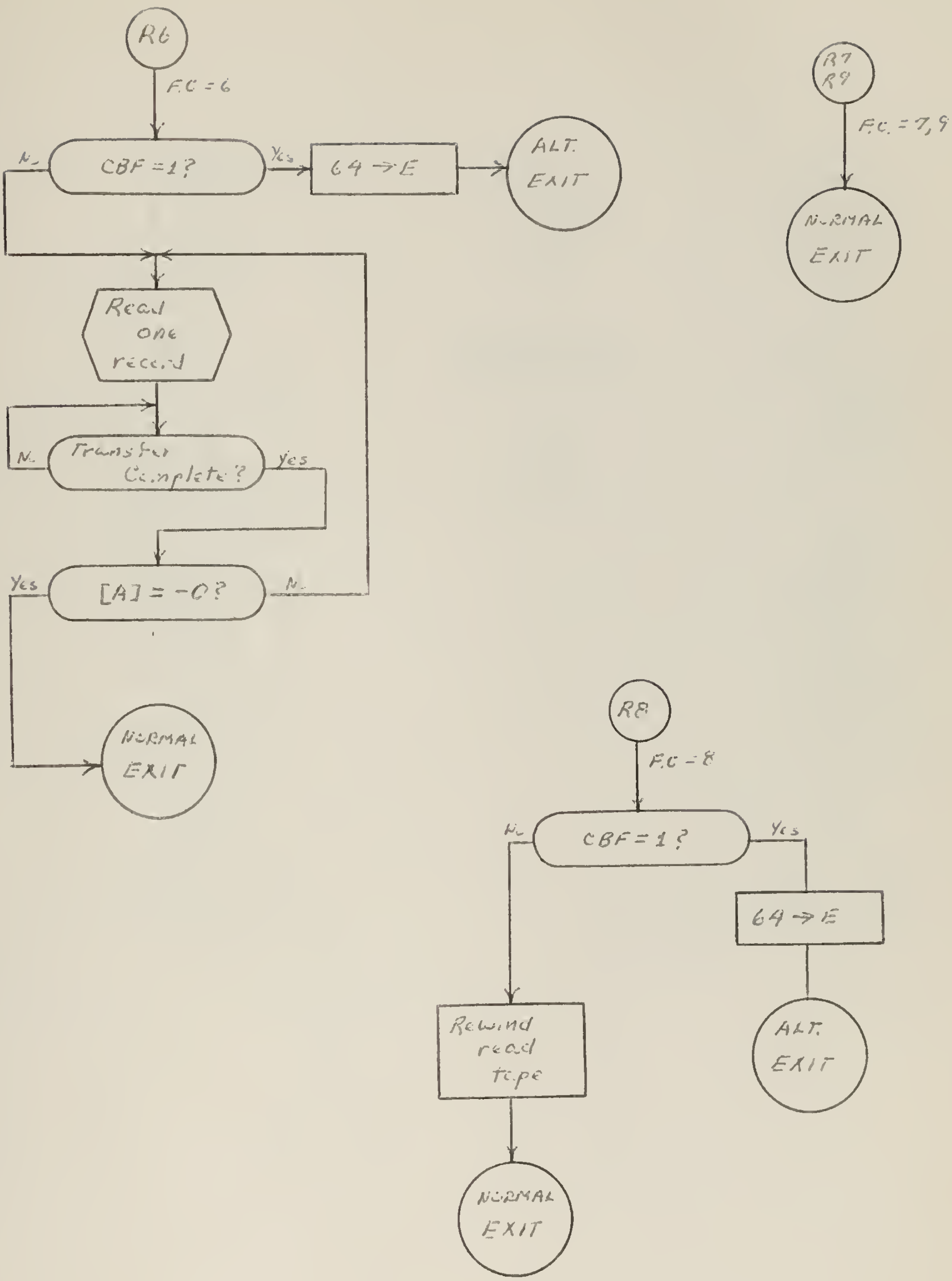

Figure 2. (Continued)

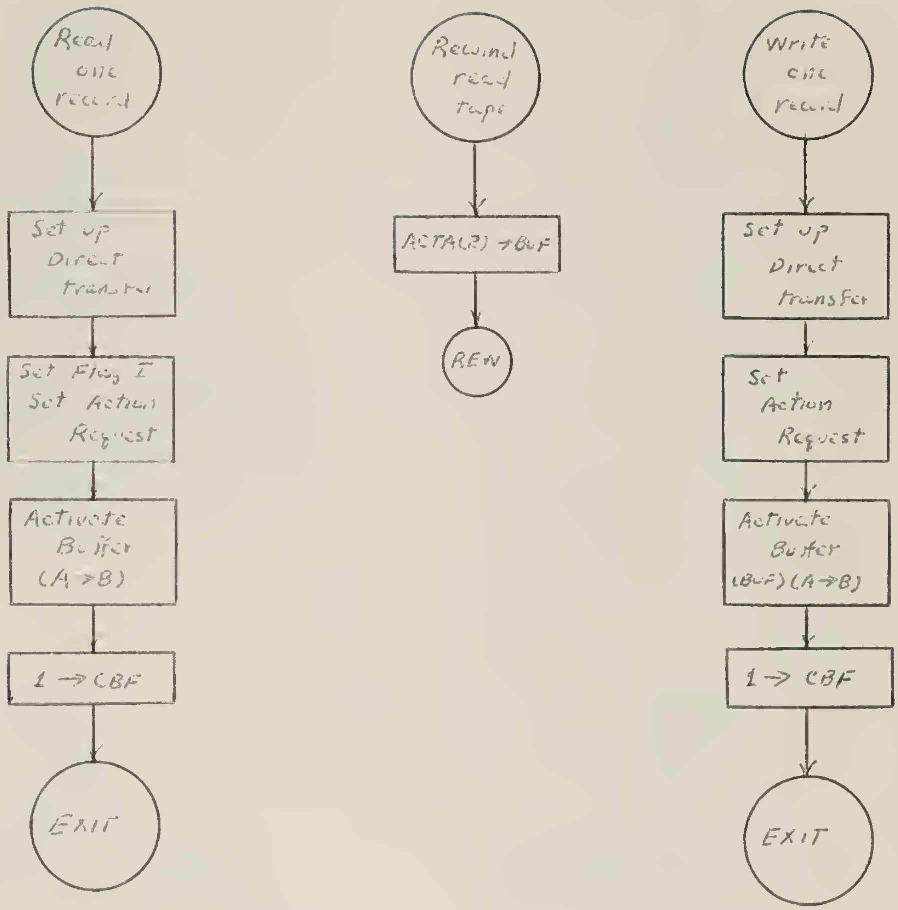

Figure 2. (Continued)

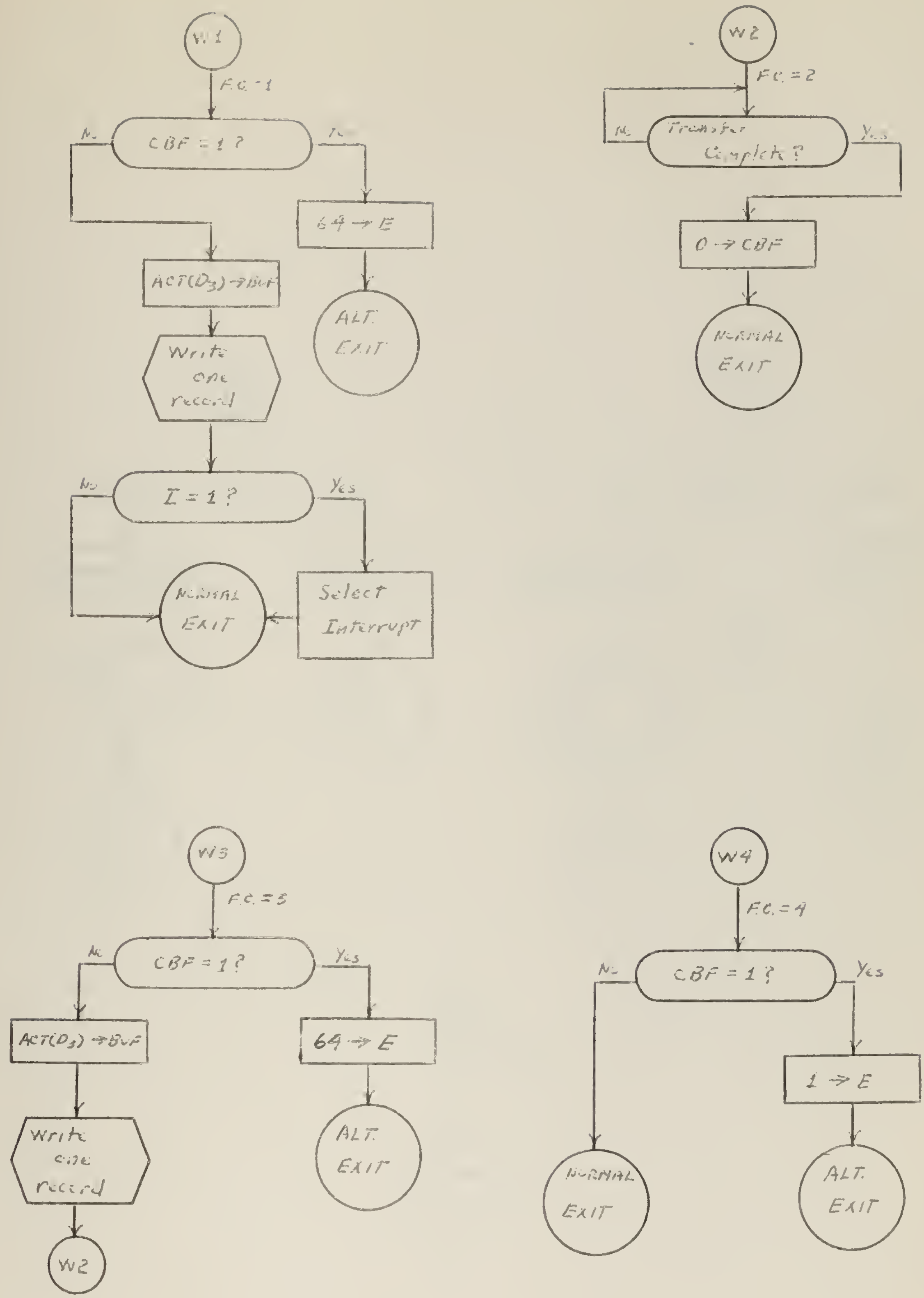

Figure 2. (Continued)

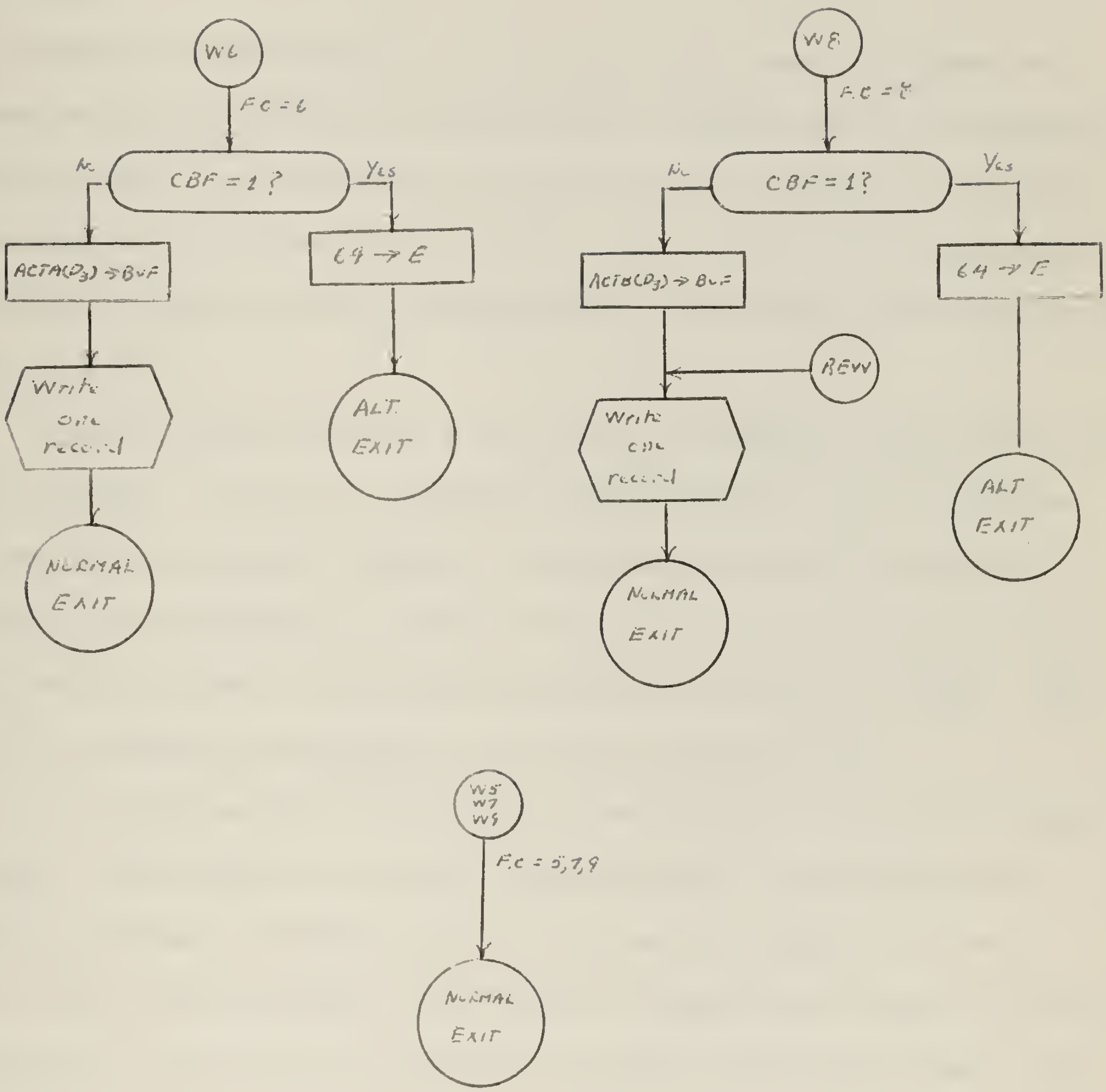

Figure 2. (Continued)

8.1.7.1 160 Satellite Read/Write Routine

The 160 computer will normally be idling in a loop waiting for an action request from the 1604. When this action request is received, FLAG I will be inspected to determine what action is required. If Flag I is NOT set, the action will be an output of the next card image from the 160. If it is set, the action will be an input of the next record from the 1604.

When the action requested is an output of the next card image, an immediate output buffer will be initiated. When this buffer is completed, the next image from the input tape will be read. When this is completed, the program will return to the wait loop. If an end-of-file mark is encountered on the input tape, the first word of the next record transferred will be a -0.

When the action requested is an input, an immediate input buffer will be initiated. When this is completed, the first word will be inspected to determine the action requested. An output buffer will be started to transfer the information to either the output tape or the printer if this has been called for. If not, the appropriate action will be taken. When this is completed, the program will return to the wait loop.

It is expected that all satellite transfers will be records of fixed length. The records from the 160 to the 1604 will consist solely of binary card images (20 1604 words). This does not imply that 10 or 15 word (80 or 120 character) records cannot be read from the input tape and transferred. They will be treated as if they were 80 column binary card images, however. The last 10 words of a 10 word record and the last five words of a 15 word record will be meaningless.

The records transferred to the 160 from the 1604 will be 120 charac-

ter BCD records. Hence, all records written on the output tape or the on-line printer will be 120 Hollerith characters in length. If a shorter Hollerith record is transferred to the 160, the record will be padded out to 120 characters with spaces before it as written on the output medium.

If a write error occurs, the tape will be backspaced and the record rewritten. This will be repeated 8 times. If the record is still not written successfully, a record of periods will be written and the information record rewritten.

A flow chart of this routine is shown in Figure 3 on the following pages.

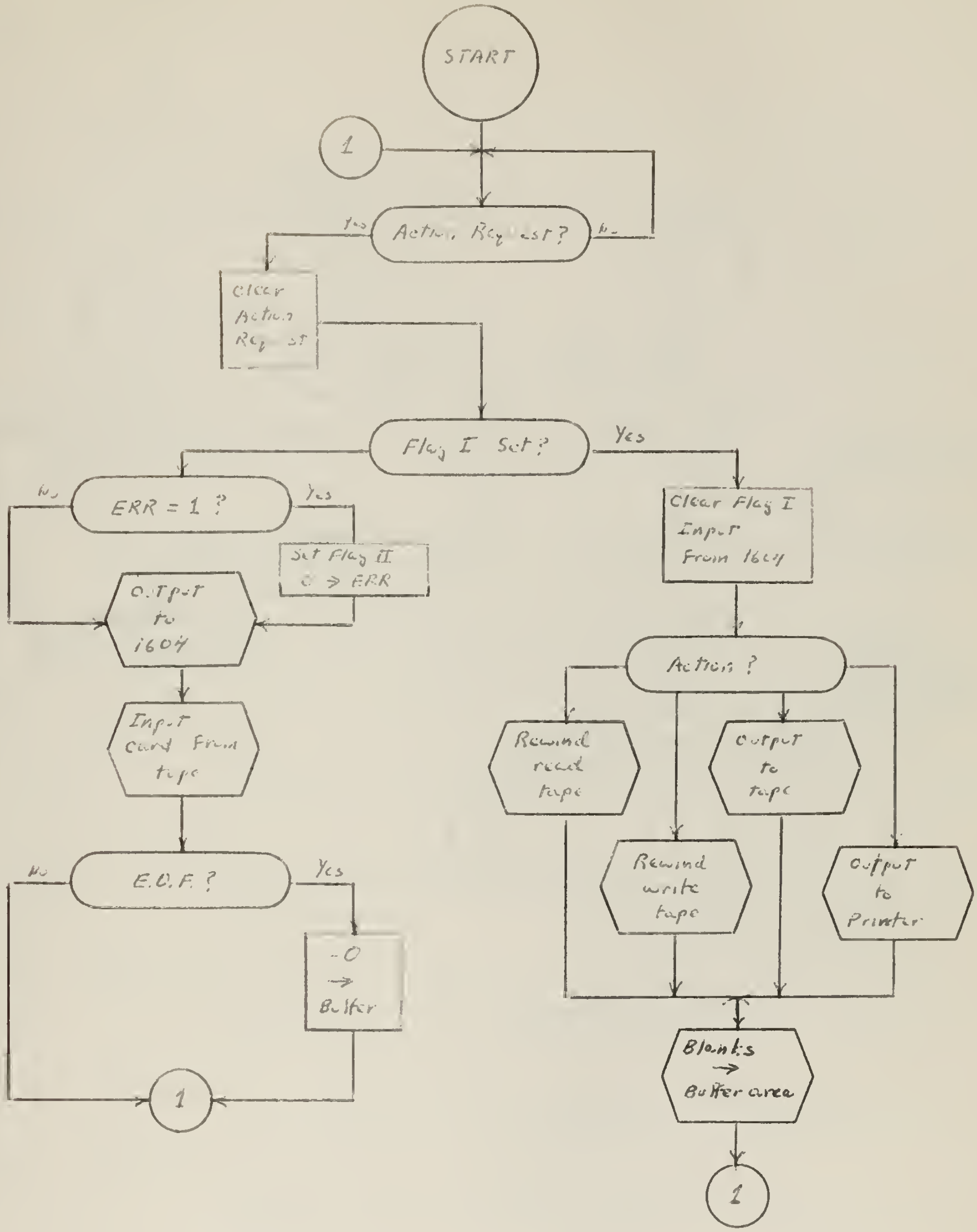

Figure 3. 160 Read/Write routine.

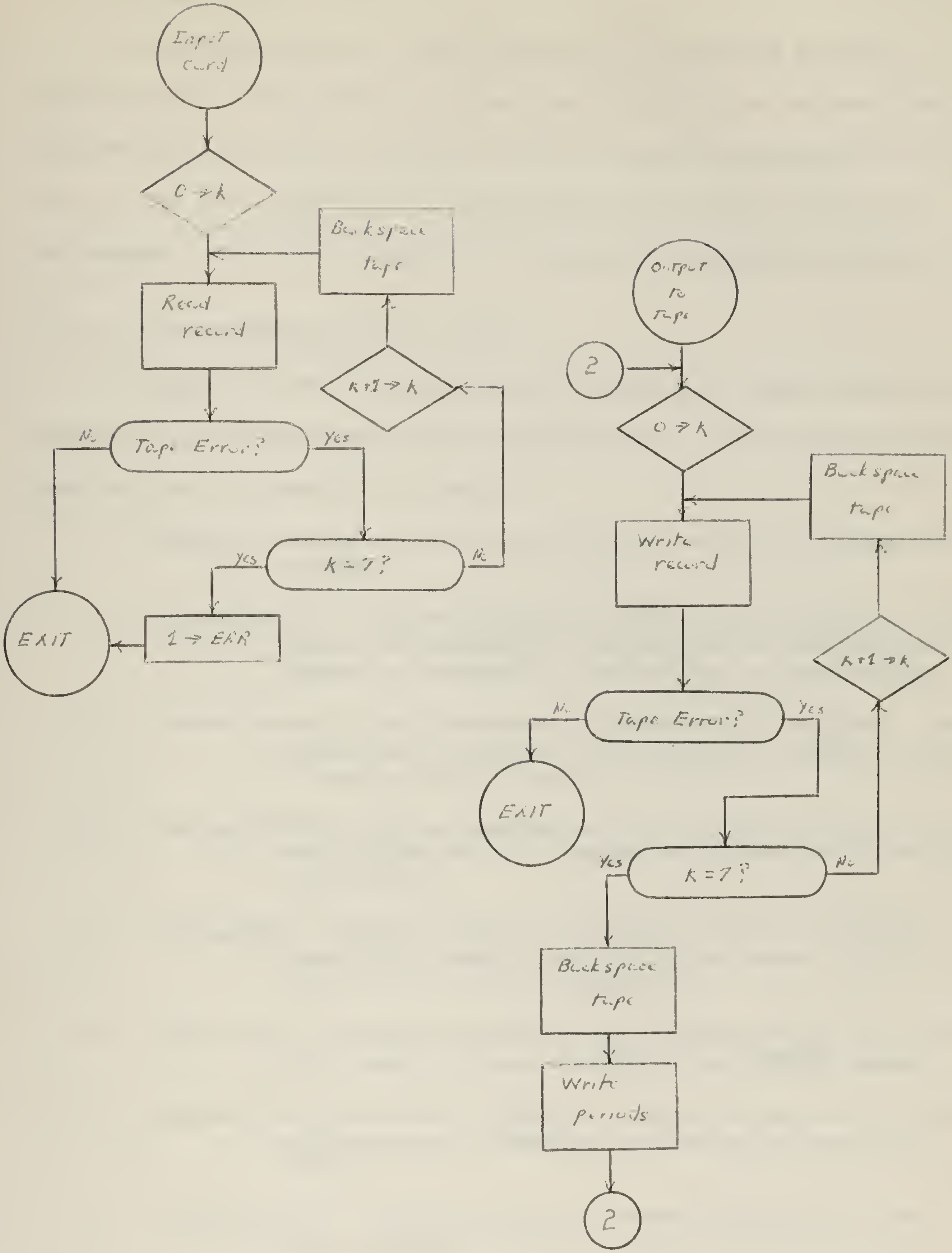

Figure 3. (Continued)

8.2 System Monitor Details

The following sections contain details of resident SM routines. These routines will always be in memory and should not be reloaded each time the LM cycles to the next job. Their use has been explained in Section 7. For more information on the remote console and the details of programming for it, the thesis by C. G. Lawson /4/ should be consulted.

8.2.1 Read Remote Keyboard Details

The flow chart shown in Figure 4 on the following pages presents the details of the Read Remote Keyboard routine (RRK). This routine is broken down into the following subroutines.

1. Control - Determines which subroutine is needed to process new character.

2. Process ENTER - Entered whenever an ENTER code is received. Completes last parameter if necessary and executes called routine if available. Sets routine for new message.

3. Process comma - Entered whenever a comma code is received. Completes last parameter if necessary and advances SYMSTO to receive next parameter.

4. Process ERASE - Entered whenever an ERASE code is received. Resets routine for new message and clears present message from display.

5. Process BKSP - Entered whenever a backspace code is received. Clears last symbol from display and resets SYMSTO to replace this symbol with new one.

6. Process CHAR - Entered whenever a code received which is not one of the above. Packs characters into SYMSTO words.

7. Convert number and store - Converts numeric parameters to binary with necessary conversion and stores the result in SYMSTO.

8. Convert number - Determines type of number and converts from BCD to binary.

9. Display new symbol - Packs last symbol received into DSPSTR so that it may be displayed.

10. Set DSPSTR - Sets new memory address and starting position of display line and clears DSPSTR.

11. Clear DSPSTR - Clears DSPSTR by setting all words except first equal to zero.

12. Output DSPSTR - Transfers DSPSTR to remote display memory. The entire DSPSTR area is transferred each time.

To aid in understanding these charts the following symbols are defined:

AVROA - Available routine addresses

AVROA(k) - Indicates k^{th} address in AVROA table

AVRON - Available routine names

AVRON(k) - Indicates k^{th} name in AVRON table

CR - Carriage return

D - Indicates number of lines currently being displayed.

DELTA - When added to DSPSTR(0), increments display memory address and Y coordinate to start next line.

DSPSTR - Storage for display words as symbol strings

DSPSTR(np) - Indicates p^{th} character in n^{th} display word

DSPSTRO - Initial DSPSTR(0). Contains initial display memory address and start of first line coordinates.

DISPZERO - Used to fill DISPSTR with null characters. Contains all zeros except for display control bits.

NUMFLAG - Set indicates parameter is a number

STOFLAG - Set indicates input number requires two words to hold in BCD.

SYMSTO - Storage for input messages

SYMSTO(j) - Indicates j^{th} word of SYMSTO

S - Current input symbol

REPEAT - Set indicates ENTER received but nc new message

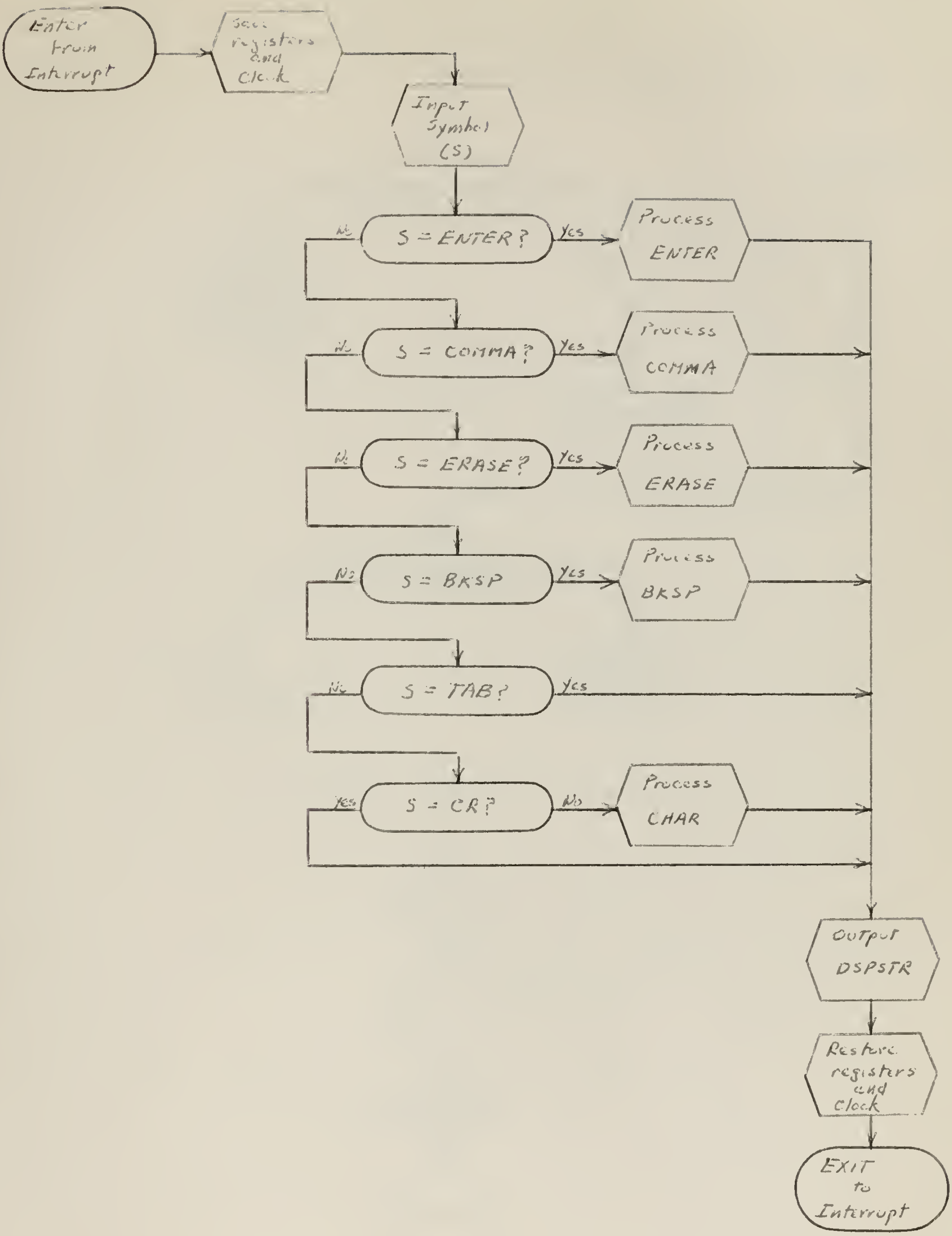

Figure 4. Flow chart of Read Remote Keyboard routine.

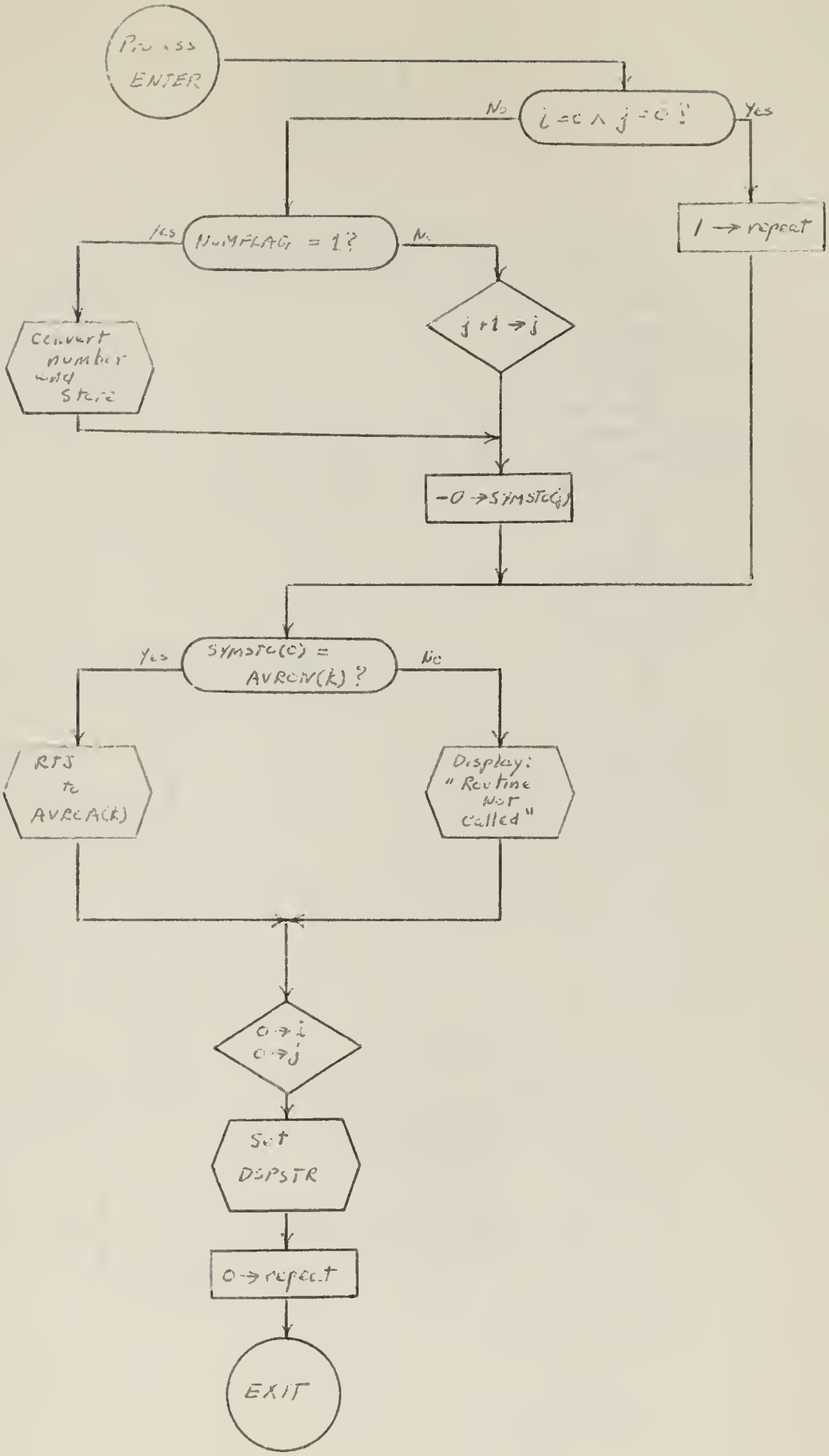

Figure 4. (Continued)

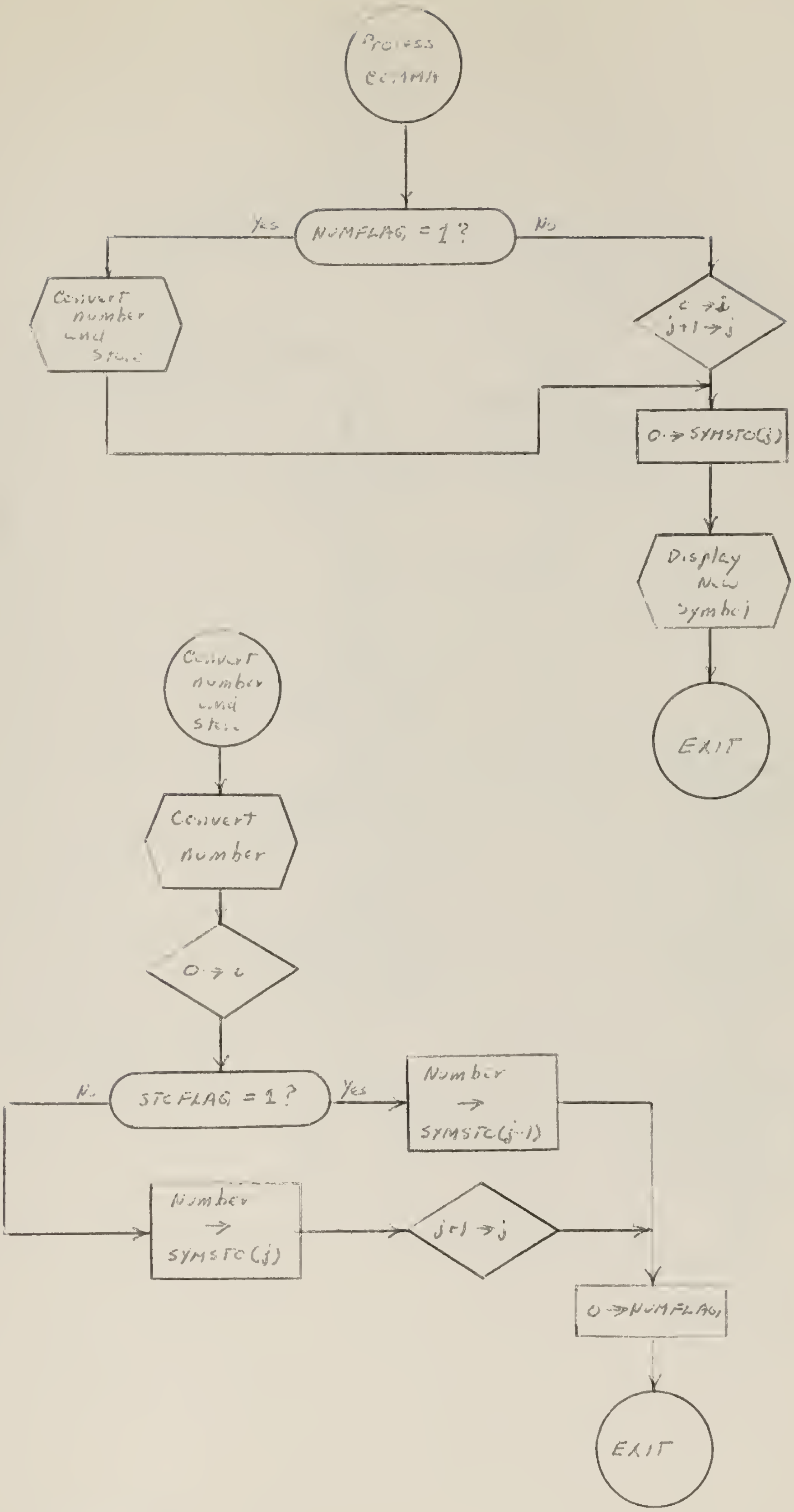

Figure 4. (Continued)

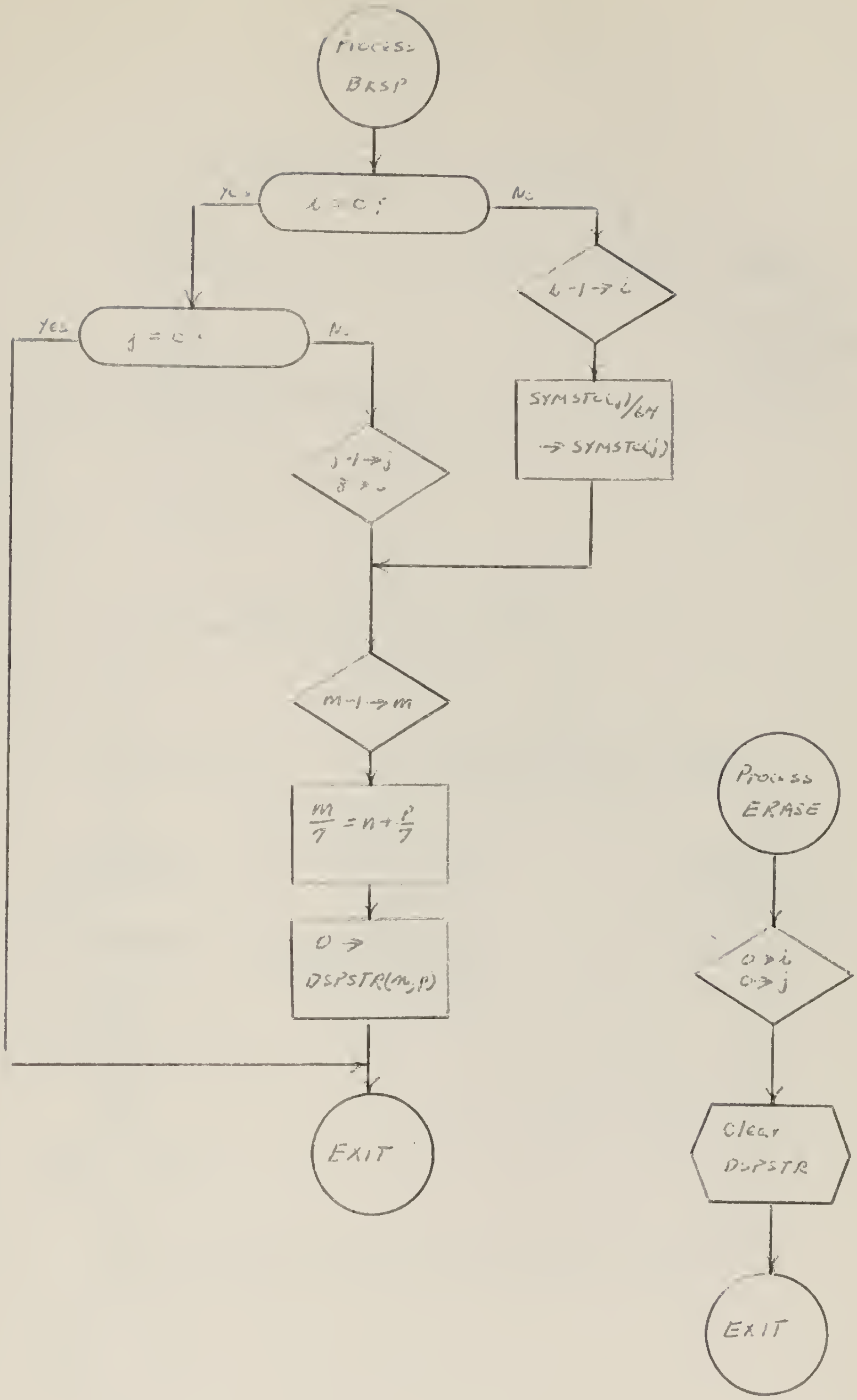

Figure 4. (Continued)

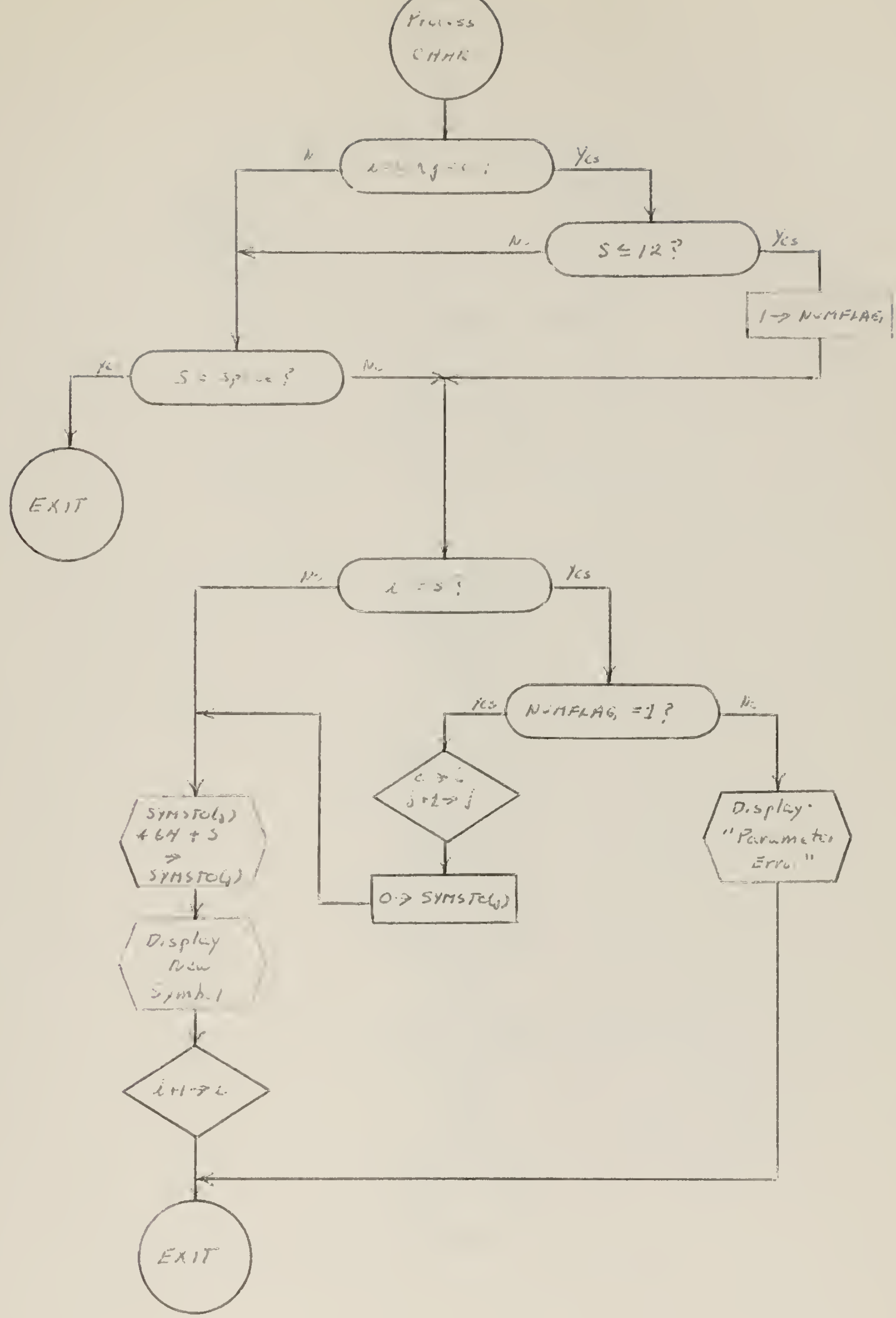

Figure 4. (Continued)

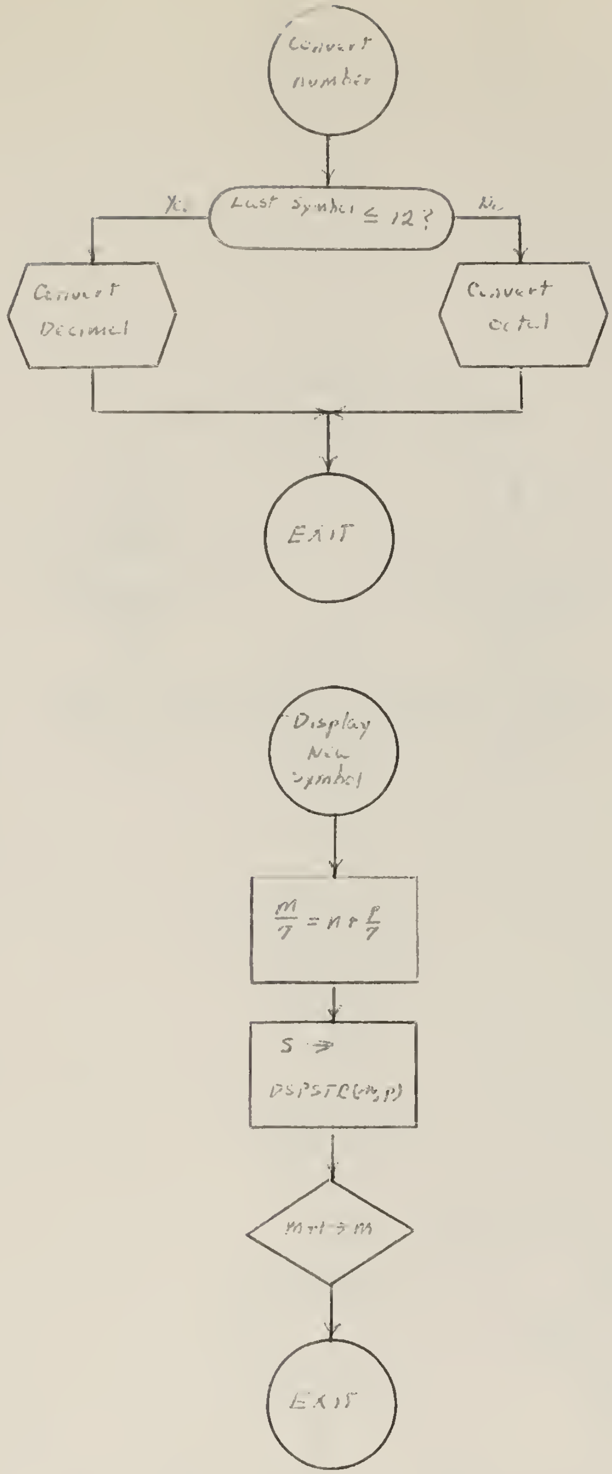

Figure 4. (Continued)

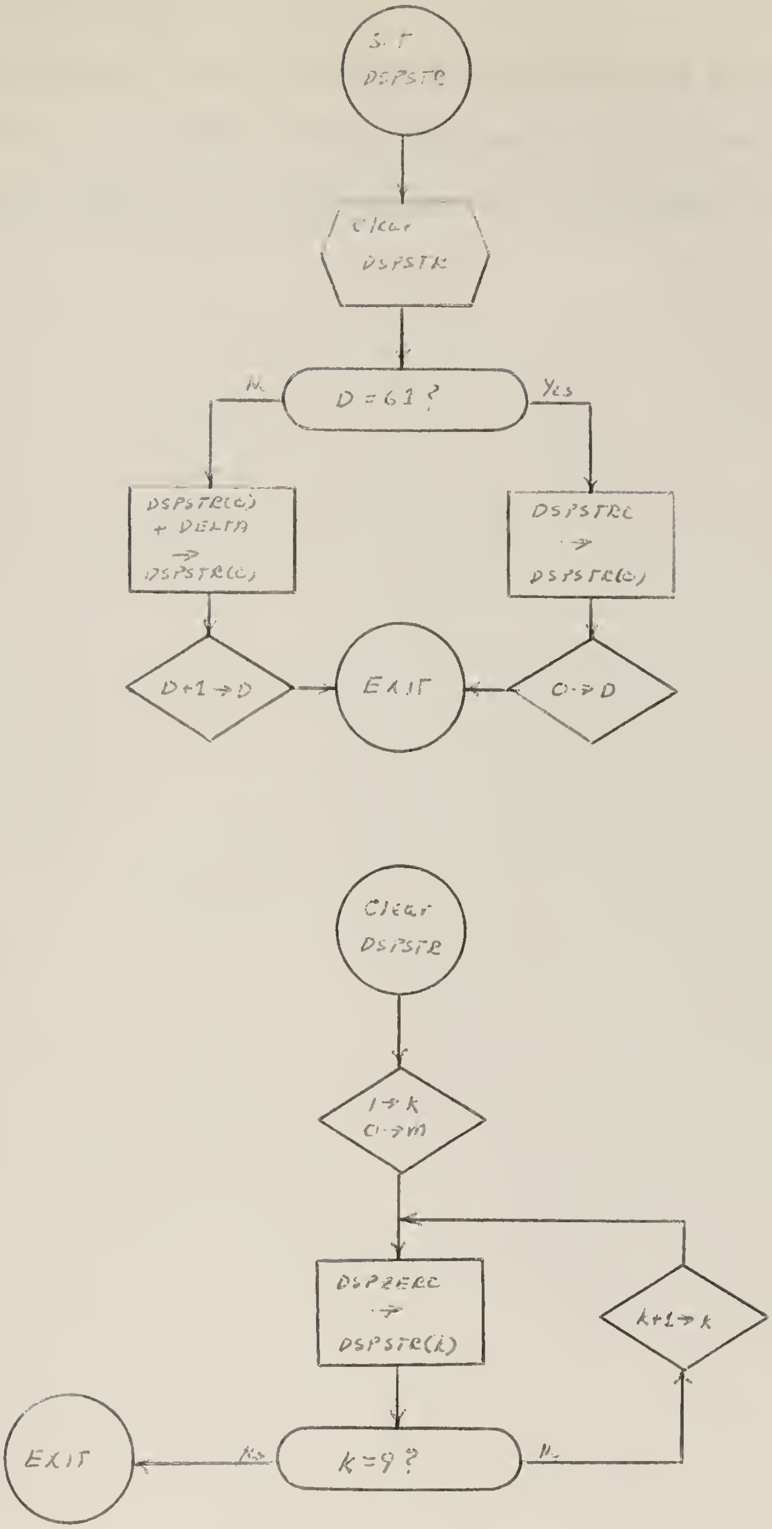

Figure 4.   (Continued)

8.2.1.1 Clear Routine

The flow chart shown in Figure 5 on the next page presents the details of the clear routine. The symbols used on this flow chart have already been defined in the description of the Read Remote Keyboard routine.

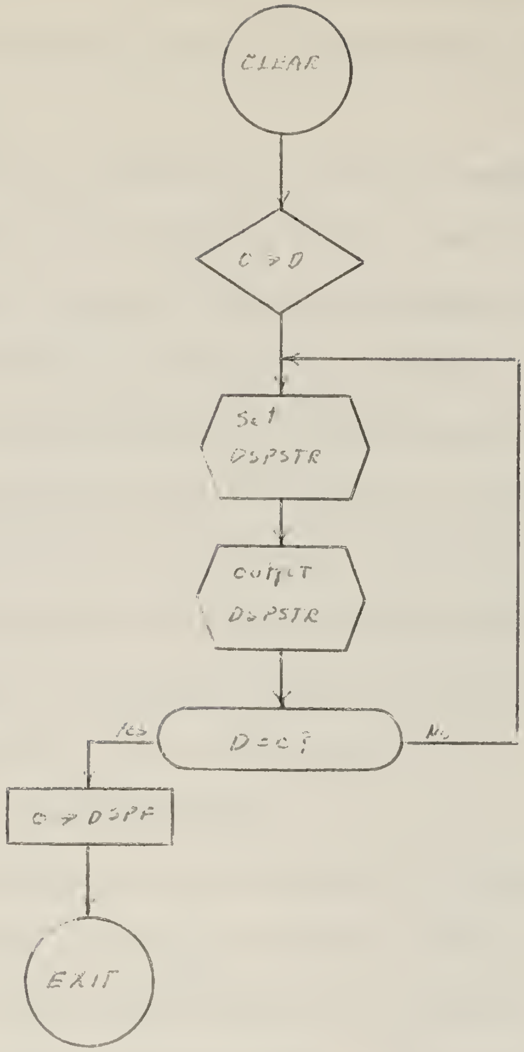

Figure 5. Flow chart of Clear routine.

8.2.2 Load Programs Routine

The flow chart shown in Figure 6 on the next page presents the details of the Load Programs routine. Three blocks on this flow chart require amplification.

1. Change RBL exit address to 1. This is required if the LM is using the RBL routine when called by the SM. The exit address of the RBL routine is changed to a point in the Load Programs routine and an exit back to the RBL routine made through the interrupt routine. When the RBL has completed its current task, it will then proceed to the Load Programs routine and its execution by the SM routine can proceed. When the SM has completed its task, it will exit directly to the LM instead of to the interrupt routine if this exit address has been changed.

2. Pack APARA - The parameter given to RBL in the A register on entrance is called APARA. It is in the form:

300FFFFFXX0LLLLL

where FFFFF is the address of the first word of available memory, LLLLL is the address of the last word of available memory and XX is the RHT entry position. A cell called MEM in the SM resident is set aside to hold the current memory available limits. The upper address of this cell (SMC_0) is the address of the first word while the lower address (SMC_1) is the address of the last word. These are entered in the APARA along with the RHT entry given in PARA1.

3. On return, RBL supplies the new limits of available memory. These limits are saved in the cell called NMEM.

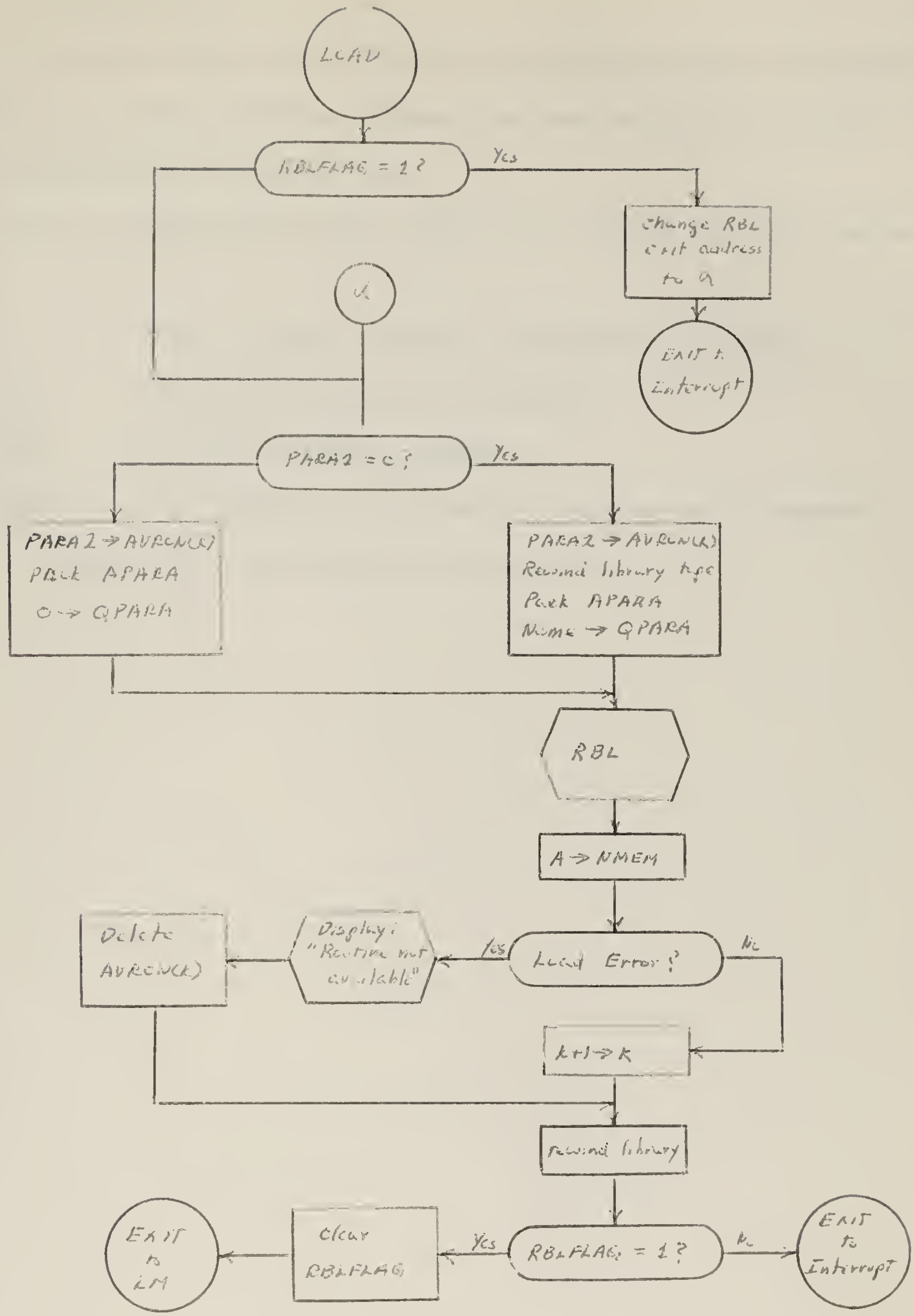

Figure 6. Flow chart of Load routine.

8.2.3 Display Routine

The flow chart shown in Figure 7 on the following pages presents the details of the Display routine. Subroutines used within this routine have been described in Section 8.2.1.

To aid in understanding these charts, the following symbols are defined:

B	- Bias. Address of start of SM portion of memory.
DSP	- Temporary storage for parameters.
DSPA	- Initial parameter storage.
DSPFLG	- Set indicates second word of line has been processed.

Others have been defined in Section 8.2.1.

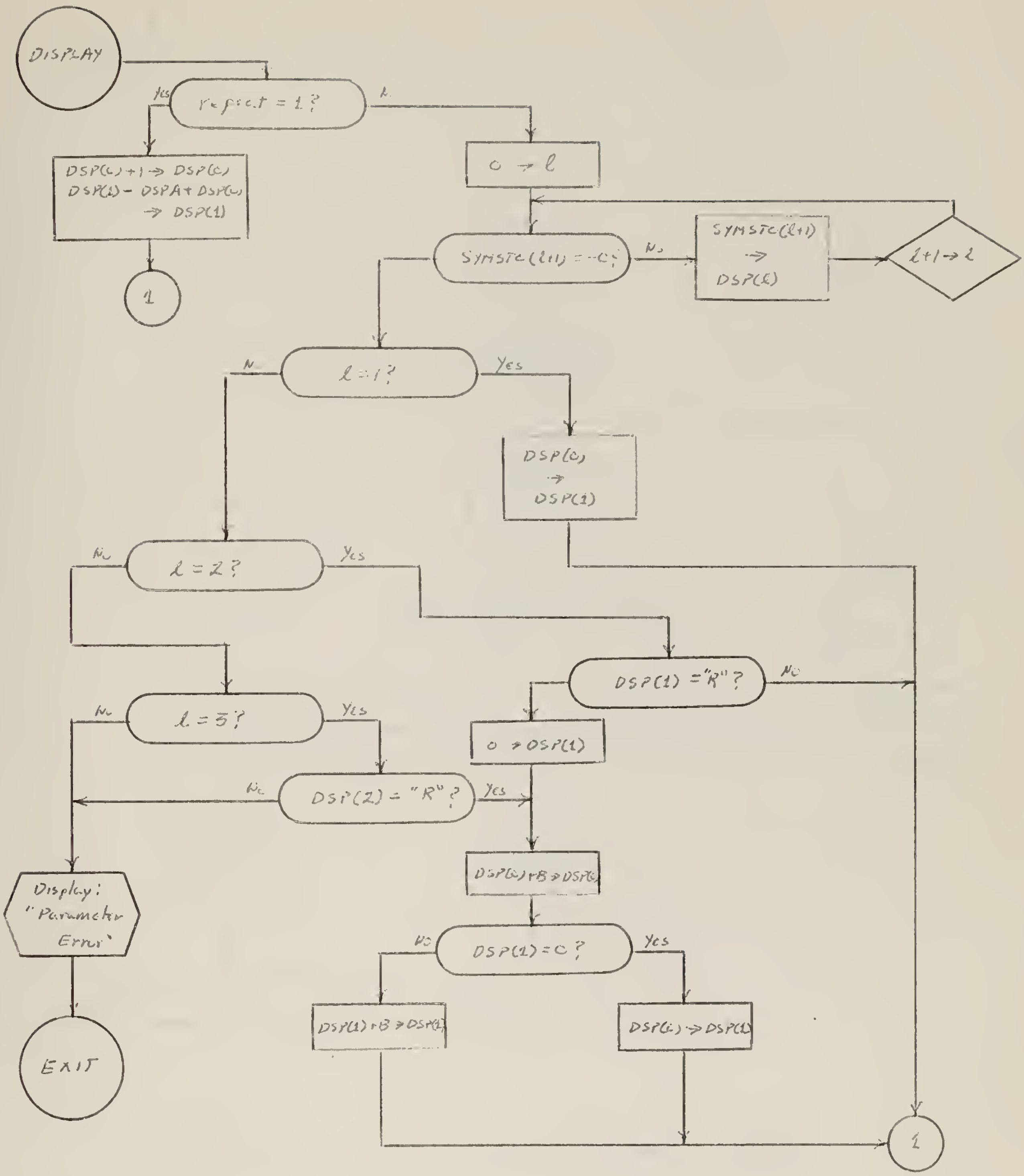

Figure 7. Flow chart of Display routine.

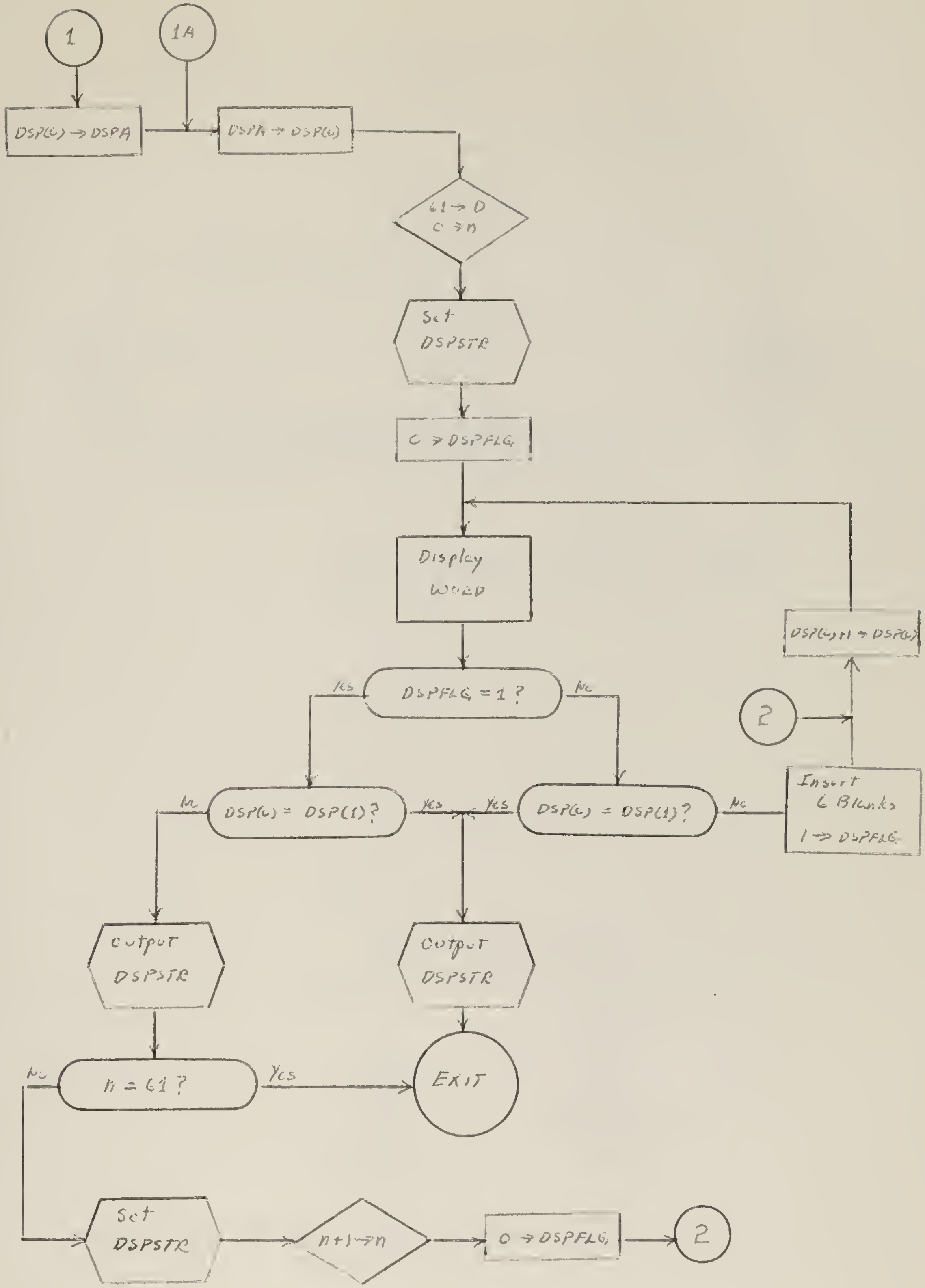

Figure 7. (Continued)

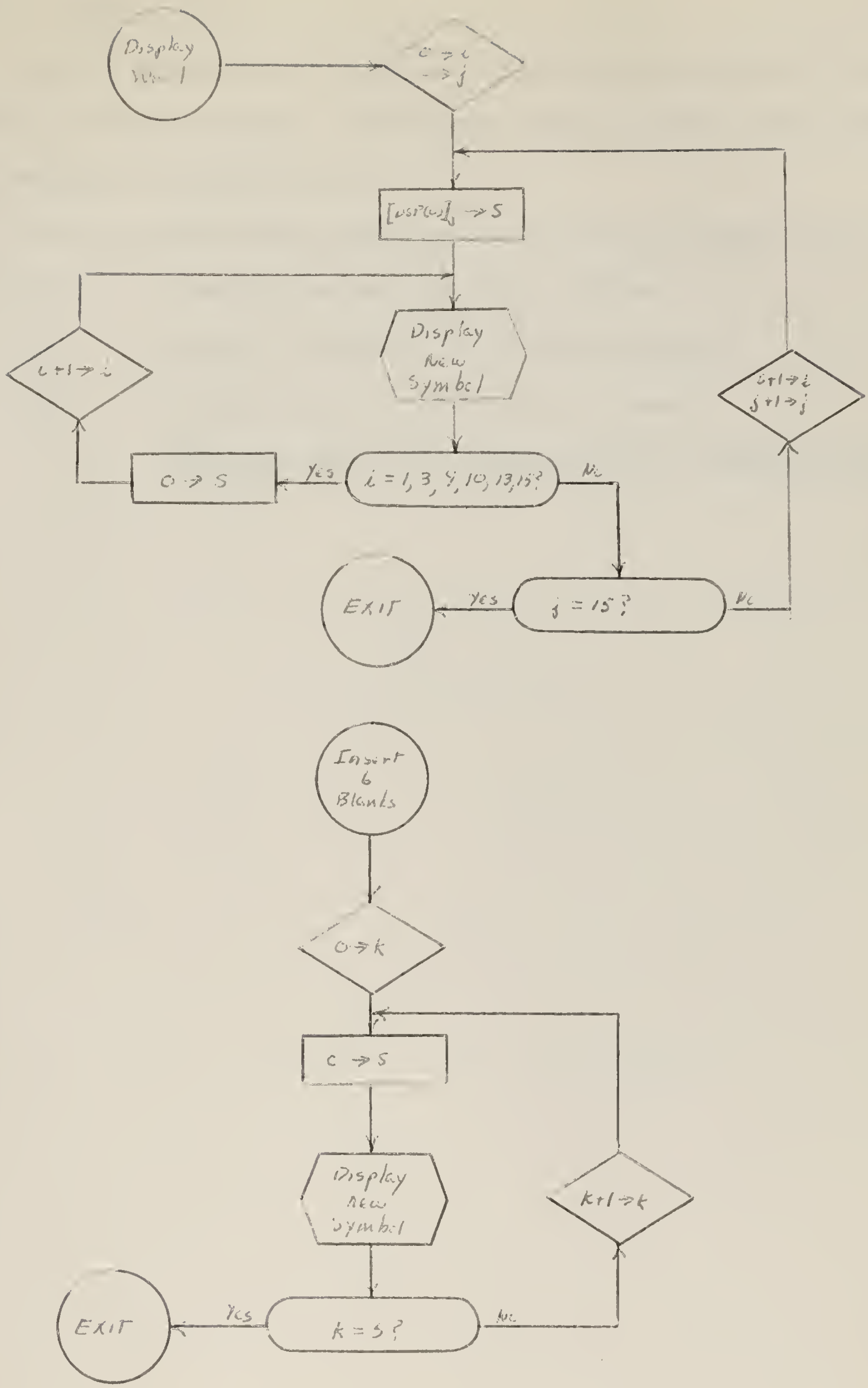

Figure 7. (Continued)

8.2.4 Enter Routine

The flow chart shown in Figure 8 on the next page presents the details of the Enter routine. Subroutines used within this routine have been described in Section 8.2.1.

To aid in understanding this chart, the following symbols are defined:

ENT - Temporary storage for Enter routine

SMC_0 - Address of start of SM portion of memory

SMC_1 - Address of end of SM portion of memory

(SMC_0 and SMC_1 are in upper and lower address respectively of MEM)

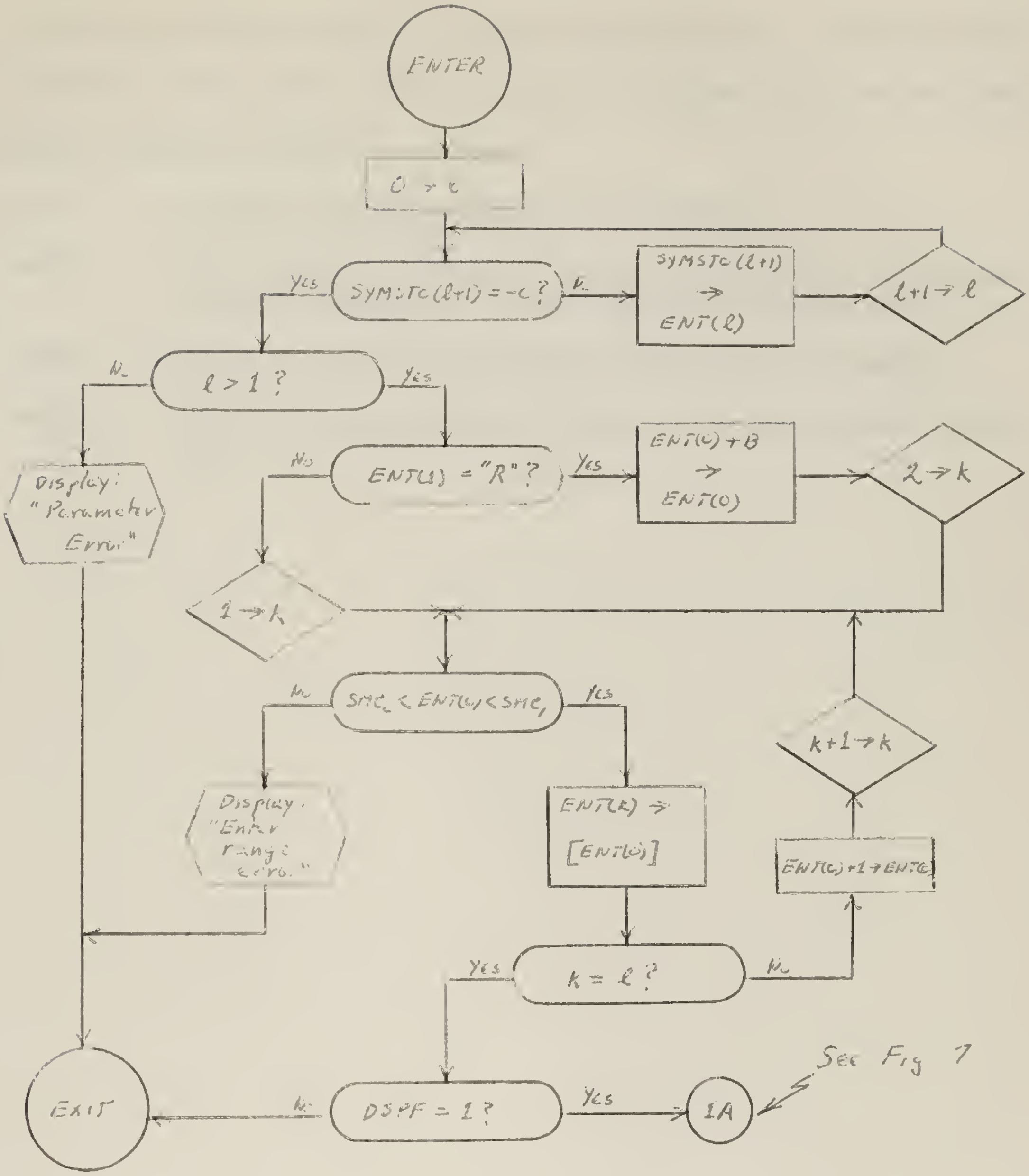

Figure 8. Flow chart of Enter routine.

8.2.5 Miscellaneous Routines

Three SM resident routines have not been described. These are HOLD, FREE, and SIZE. Their flow charts are given in Figure 9 on the next page. The symbols used are defined as follows:

MEM	- Current effective limits of SM memory.
BMEM	- New limits of SM memory set by SIZE routine. Bootstrap Loader transfers it to MEM when LM cycles to next job.
IMEM	- Initial limits of SM memory when system is loaded.
NMEM	- Memory limit parameter returned from RBL on last use by SM.

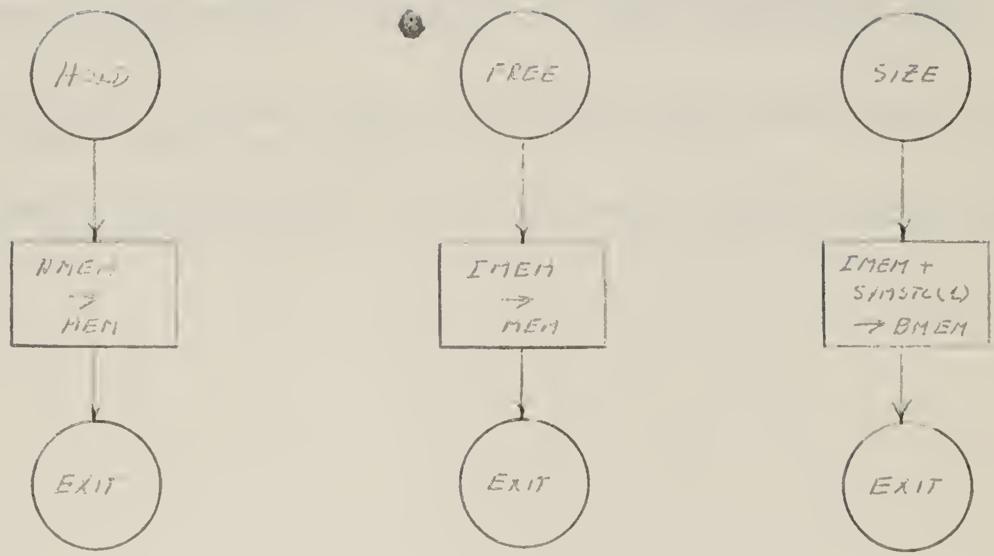

Figure 9. Flow charts of Hold, Free and Size routines.

BIBLIOGRAPHY

1. Teague, H. M., "Real-Time Time-Shared Computer Project", Communications of the ACM, 5, 1 (January, 1962) Research Summaries, 62.

2. Corbato, F. J., "An Experimental Time-Sharing System", Proceedings, Spring Joint Computer Conference, 1962, 335.

3. Mitchell, R. W., Specifications for the Control Data 1604 CO-OP Monitor System, 9 March 1962, unpublished.

4. Lawson, C. G., "An Integrated Display and Control System for Man-Machine Communication", MS Thesis, U. S. Naval Postgraduate School, Monterey, Calif., 1962.

5. CO-OP Monitor Operator's Guide, Field Test Issue, Control Data Corporation, Minneapolis, Minn.

6. Fortran-62 Reference Manual, Field Test Issue, Control Data Corporation, Minneapolis, Minn.

APPENDIX A

This appendix is a summary of the characteristics of the CO-OP Monitor System. This material was obtained from the now existing reports of the system /3,5,6/. It should be noted that /3/ is a preliminary report of the System. It has not been appended in its entirety due to its length.

A1. Introduction

In an attempt to get universal agreement on the specifications for the CO-OP Monitor System, the following design criteria were established.

1. The resident portion of the monitor system should be kept to a minimum by including in resident only those functions which are absolutely essential for continuous job processing. Other useful functions sometimes included in monitors would be callable only when needed from the library.

2. All but one tape unit, namely the library tape, would be available to the programmer.

3. The monitor system and all programs run under the monitor system would be completely independent of the kind of input-output equipment attached to a particular system except to the extent that an environment table would be supplied to the monitor defining the I/O media and the names of library subroutines capable of driving the I/O equipment involved.

4. Other monitor systems considered as sub-monitors may operate within the basic CO-OP Monitor System.

5. The monitor would be designed in such a way that the possibility of including satellite control routines would not be precluded.

6. The monitor would be equally effective for both scientific and business applications.

A2. General description

The CO-Op Monitor is a multi-level control system designed to process jobs automatically with a minimum of operator intervention.

The first level of the system, the Master Control System, (MCS), provides for:

1. Automatic sequencing of jobs including accounting records of all jobs processed by the monitor.

2. Operator-machine communications

3. Memory allocation

4. Assignment of Input/Output equipment

5. Linkage to I/O media

6. Universal and flexible linkage to the 1604 interrupt feature

7. Loading of program cards and library routines

8. Recovery procedures in the event of program failure

9. Linkage to second level control systems.

Any routine on the library tape may be called as a second level control system. These second level control systems may be simple single phase monitors, more complex multi-phase monitors, or parallel programming systems. As the need arises, a second level system may be added by a simple edit pass at the library tape adding the system to the standard "subroutine" library.

One second level system will be included in the basic CO-OP Monitor package called the CO-OP Control System. The CO-OP Control System will provide linkage to third level systems such as FORTRAN, CODAP, COBAL, etc. Any program to be executed under control of the CO-OP Control System may also be considered a third level system.

The library tape consists of two physical files. The first file

contains the Master Control System, a Bootstrap load routine for loading all or portions of the Master Control System, and a recovery program. The routines in the first file are in absolute binary form. Included in the Master Control System will be the I/O routine for the particular equipment on which the library tape is to be mounted.

The second file of the library tape contains a directory and the subroutine library. Records in this file are in relocatable binary. Included in the subroutine library are Subordinate Control Systems, Input/Output Routines, compilers, assemblers, and utility routines.

The following is the normal sequence of operations of the CC-OP Monitor System.

1. A bootstrap sequence will cause the first record of the library tape to be loaded.

2. Control is given to bootstrap routine which skips over the recovery record and loads MCS.

3. Control is given to MCS which initializes memory and indicates to operator that the system is ready.

4. Operator signals to begin processing.

5. MCS jumps to Relocatable Binary Loader requesting I/O routine for standard input medium.

6. MCS reads IDC card of job, logs it on and initializes the clock.

7. Control is given to Equipment Assignment routine which generates a Running Hardware Table.

8. MCS frees all but portion of low memory.

9. Control is given to Relocatable Binary Loader which loads all I/O routines implied by the Running Hardware Table and the Control System specified on the job card.

10. Control is given to Control System which controls remainder of job.

When processing of a job has been completed, control is returned to MCS with an indication that the job is to be terminated "normally" or "abnormally". In case of "normal" termination, control is given to the bootstrap load routine which rewinds the library tape, reloads MCS, logs off the job, and logs on the next job. In case of "abnormal" termination, the recovery key which was specified on the ID card is examined. If the option requested involves only a console scoop, this is given and normal termination procedure ensues. If the option requested involves a memory dump, the library tape is rewound and the full recovery program loaded by the bootstrap load routine on top of a portion of resident which is not needed for the dump. After the dump has been completed, control is again given to the bootstrap load routine and "normal" termination ensues.

The extent to which the running program or a Subordinate Control System is controlled by MCS is logging on, logging off, and recovery in case of program failure. Also MCS provides links to I/O routines, an interrupt package, a program and library subroutine loader, etc. Thus with very few restrictions, a subordinate control system may be designed to operate under MCS and take advantage of the facilities provided.

A3. The Master Control System contains the following sections:

1. Job sequencer
2. Equipment assignment
3. Operator control
4. Clock control
5. I/O package
6. Interrupt package
7. Relocatable Binary Loader
8. Boostrap loader

A brief description of each of these follows with only enough detail to understand their purpose.

A3.1 Job Sequencer

The job sequencer is initialed by an operator statement. The initiation sequence is as follows:

1. Read operator ID card and job card.
2. If this is not the first job of a batch, log the previous job off.
3. Set clock to cause interrupt in the event that the time indicated on the job card should elapse.
4. Write operator ID card and job card on operator comment, standard output and accounting medium.
5. Set available memory to RTJ COOPERRX.
6. Assign I/O units and load necessary I/O routines.
7. Transfer output limitation and recovery key to the appropriate routines.
8. Call and initiate control routine.

Upon job termination, the sequencer will remove and clear all in-

terrupts, clear all arithmetic faults, rewind all tapes designated scratch on the job card, and rewind with interlock all input and output tapes designated on the job card, and skip to the next job.

Processing will be terminated by an END MONITOR INPUT record on the input medium or by an operator statement.

The job card will contain the following information:

1. Control routine
2. Account number
3. Programmer's name
4. Logical I/O units
5. Maximum number of minutes
6. Maximum number of lines on standard output medium
7. Recovery key
8. Comments

A3.2 Equipment Assignment

This section of the Monitor is concerned with the reservation of memory and allocation of input/output units to programs and/or other sections of the Monitor. The reserving and releasing of memory is performed by the Memory Recorder.

The available memory will always be maintained as one continuous block. Memory may be reserved from either end. A call for release of memory may be made at any time but it will become effective only when available memory is adjacent to it. The Memory Recorder will normally be used only by the Monitor in loading programs and tables.

All references for input and output under the Monitor will be made by logical unit number. This routine assigns physical units to the logi-

cal units and is concerned with three I/O tables.

First is the Available Equipment Table (AET). This table contains a list of all equipment available for use during Monitor execution. Also, in this table are the assignments for the standard I/O units.

The second table is called the AEDNT (available equipment driver name table) and consists of two words per entry. The first word of an entry contains the name of the I/O subroutine. The second word of an AEDNT entry contains either zero or the relocated location of the particular I/O subroutine if it is in memory.

The third table is the Running Hardware Table (RHT). An RHT is generated for each job and consists of AET entries with the entry address for the appropriate routine and an indication as to the allowable usage for this job. By use of the equivalence subfield on the job card, one physical unit may be assigned to more than one logical unit. Entries 1 through 49 may be defined according to the wishes of the programmer. The standard I/O entries have been defined in Section 6.3.

A3.3 Operator Control

This section of the Monitor will be used for all communication between the operator and the monitor. Communications are always via the typewriter. The operator may call for execution of given routines by typing the appropriate control statement. The control statement consists of the name of the routine followed by its arguments. Monitor to operator messages may be given by using a write on the standard output comment medium (logical unit 54).

A3.4 Clock Control

In operating the clock under a monitor system, there are two func-

tions which must be performed:

1. Job sequencer timing - this includes imposing an overall time limit on the job and maintaining an accurate lapsed time record.

2. Programmer timing - this includes reading the lapsed time and setting the clock for interrupt after a given time.

Since the programmer may wish to impose separate time limits on the individual loops of a loop hierarchy, the clock routine will be capable of setting a limit within a limit. The clock routine will also maintain a counter for the lapsed time for this job and a cell containing the value for the current setting of the real-time clock. Each time the clock routine is entered with a new time limit, the table of remaining time for each setting and the lapsed time counter will be updated. If, at any time, the real-time clock is discovered to be destroyed, the job will be terminated.

A3.5 I/O Package

The I/O package for the Monitor consists of two subroutines, READ and WRITE, which drive individual subroutines, one for each type of equipment on the computer. The function of the individual I/O routines is to transmit one physical record or n words, which ever occurs sooner. In addition, the routine must perform all auxiliary operations connected with transmission of data including checking for errors. Each subroutine has a single entrance and accepts a calling sequence which is standard for all the I/O routines. The function desired is specified by a function code in the calling sequence. There are nine possible function codes and each is meaningful to each of the I/O subroutines. Two methods for buffering information are provided. The first of these makes use of the sense in-

structions and the second takes advantage of the interrupt feature.

A3.6 Interrupt Package

The interrupt feature of the 1604 forms the basis of the interrupt package. The purpose of this package is to simplify the use of this feature for the user. It is divided into three parts. Interrupt selection (SELECT), interrupt detection (DETECT) and interrupt removal (REMOVE).

SELECT accepts a calling sequence from the user which specifies the type of interrupt desired and the location of a closed subroutine (user provided) which is to be executed when the interrupt occurs. Its primary function is to select the requested interrupt. It must also provide DETECT with the information necessary to determine the cause of the interrupt and take the appropriate action.

DETECT is entered at the time an interrupt occurs. Using the information provided by SELECT it determines the cause of the interrupt, removes the interrupt selection to insure against a second interrupt from the same cause, executes the appropriate user provided subroutine and returns to the main program via location 7.

REMOVE accepts a calling sequence which specifies the type of selected interrupt which is to be removed. It removes the interrupt selection and also removes from DETECT all reference to this selection.

The CO-OP Monitor has need to use two of the interrupts. These are the interrupt on carriage return (used for operator communication) and the interrupt on arithmetic fault (used for clock control).

A3.7 Relocatable Binary Loader

The Relocatable Binary Loader will be used to load a program comprised of one or more subprograms and any number of library subroutines.

The form of the library subroutines is identical to that of the subprograms. The subprograms are on relocatable binary cards or are discrete magnetic tape records of these card images.

The library subroutines will be in a specified file on a magnetic tape. The first record of each library subroutine consists of contiguous relocatable binary card images. Optionally each library subroutine may consist of more than one record, however only the first of these will be loader controlled.

The loader will be constructed in such a way that a complete program may be "segmented" into subprograms and that each subprogram may be compiled or assembled independently of the other subprograms with which it communicates. Provisions will be made in the loader for subprograms and library subroutines to share "non-local" data regions referred to as COMMON.

The general operation of the loader will be to read in and relocate subprograms and library subroutines. Control is given to the loader from the Monitor via a return jump. Control is returned to the Monitor when the loader has completed its assigned task. Then the jump is made to the loader, the first and last words of available memory are given as a parameter. When control is returned to the Monitor, the new memory limits are given. A MAP of the memory assignments made by the loader is given in figure 10.

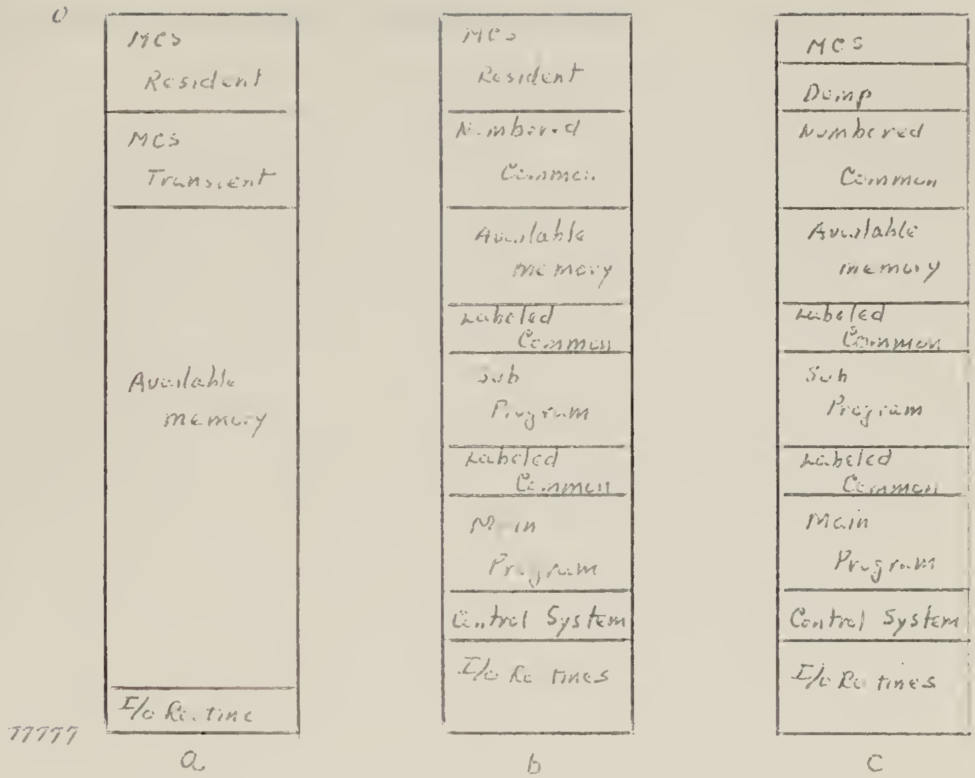

Figure 10. Memory usage under CO-OP Monitor System.
a - Initial, b - During program execution, c - During "abnormal" termination when full dump is called for.

A3.8 Bootstrap Loader

The function of the Bootstrap routine is to load, on option, and transfer control to either the MCS or the Recovery and Dump routine. The Bootstrap routine will be the first record on the monitor system tape and will include the Available Equipment Table.

A4. Subordinate Control Systems

The job sequencer of the CO-OP Monitor is for the purpose of performing the necessary steps to getting every job, regardless of its type, started on the computer. This does not include calling compilers or assemblers as required by the program or the final preparation of binary decks for loading. These jobs are the responsibility of the "Control System". The control routine is responsible for all "intra-job sequencing", ie, calling necessary compilers or assemblers, transferring binary cards to load tape, calling loader, and other operations necessary to prepare and execute a user's job.

Under the CO-OP Monitor, it will be possible for each installation to write its own control routine for the processing of control cards.

A5. CO-OP Control System

The CO-OP Control System is for the processing of intra-job control statements and also contains three internal routines for binary deck manipulation, loading and execution. Any library routine is also callable and arguments may be transmitted to the routines.

APPENDIX B

The following equipment is assumed in the design of AMPS. No modifications to the standard units are required.

1. CDC 1604 computer
2. Two CDC 1607 tape subsystems
3. Two CDC 160 computers in satellite
4. CDC 163-2 tape subsystem for 160 computer
5. DD65 display console
6. Line printer